

密级: 内部



中国科学院大学
University of Chinese Academy of Sciences

博士学位论文

基于多级异构并行处理架构的视觉芯片设计研究

作者姓名: 杨杰

指导教师: 刘剑 研究员 中国科学院半导体研究所

吴南健 研究员 中国科学院半导体研究所

学位类别: 工学 博士

学科专业: 微电子学与固体电子学

研究所: 中国科学院半导体研究所

2015 年 05 月

Design of Vision Chip Based On
Hierarchical Heterogeneous Parallel Processing Architecture

By

Yang Jie

**A Dissertation Submitted to
The University of Chinese Academy of Sciences
In partial fulfillment of the requirement
For the degree of
Doctor of Engineering**

**Institute of Semiconductors, Chinese Academy of Sciences
May, 2015**

关于学位论文使用权声明

任何收存和保管本论文各种版本的单位和个人，未经著作权人授权，不得将本论文转借他人并复印、抄录、拍照、或以任何方式传播。否则，引起有碍著作权人著作权益之问题，将可能承担法律责任。

关于学位论文使用授权的说明

本人完全了解中国科学院半导体研究所有关保存、使用学位论文的规定，即：中国科学院半导体研究所有权保留学位论文的副本，允许该论文被查阅；中国科学院半导体研究所可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

（涉密的学位论文在解密后应遵守此规定）

签 名：

导师签名：

日 期：

关于学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签 名：

导师签名：

日 期：

摘要

视觉芯片是一种旨在构建类似人类视觉系统功能的超大规模数模混合片上系统芯片。它集成了图像传感器和大规模并行处理器。它的潜在视觉处理性能超过了传统图像处理系统，并且具有功耗低和集成度高的优点。但是，目前已报道的视觉芯片的性能仍然无法满足完成高速高精度视觉处理的应用需求，尤其是在图像特征提取、目标检测、追踪等应用中凸显的计算能力不足，制约了视觉芯片的实际应用。本文结合视觉芯片架构、并行处理算法尤其是机器学习算法等开展了新型视觉芯片的设计研究，并取得了如下主要创新性研究成果：

论文提出了一种基于多级异构并行处理（Hierarchical Heterogeneous Parallel Processing, HERO）的新型视觉芯片架构。该架构集成了图像传感器、像素级、块级以及分布式多级异构并行处理器，具备图像特征检测、特征提取以及特征分类的处理功能，可高效、灵活地完成各类典型视觉处理算法。

论文提出了适合基于多级异构并行处理架构的新型视觉芯片的视觉处理专用指令集。该指令集支持像素级、块级以及分布式并行操作的编程，并设计了块处理单元电路（Patch Processing Unit, PPU）、处理单元电路（Processing Element, PE）、车道电路（Lane）以及缓存控制电路以实现指令集功能。

论文提出了可用于视觉芯片大规模并行计算的特征点检测、图像特征提取算法，其中包括最小核值相似区（SUSAN）特征点检测、FAST 特征点检测、尺度不变特征变换（SIFT）特征点检测、LBP 纹理特征描述、HoG 特征描述。论文还提出了用于视觉芯片并行分类的自组织映射神经网络和自适应提升算法。

基于上述提出的架构、电路和算法，论文采用 FPGA 实现了 HERO 处理器，并搭建了相应测试平台，测试结果显示 HERO 可以实现高性能的特征检测、特征描述和特征分类算法，在 50MHz 系统时钟下达到了 64GOPS 的峰值运算性能，超过了已报道的其他视觉芯片。

关键词：视觉芯片；异构；并行；特征分类；计算机视觉；机器学习；

Abstract

Vision chip tries to build a very large scale mixed signal system on chip that has similar function as human visual system (HVS). It consists of image sensor and massive parallel processor. It has shown great potential in visual processing when compared with traditional image processing systems. It has advantage in aspects of power consumption and system integration. However, the performance of the reported vision chips still cannot satisfy the requirements of high-speed and high-precision visual processing applications, especially in image feature extraction, object detection and target tracking, it greatly limits the application of vision chip. This thesis carry out the research of novel vision chip design by combining vision chip architecture, parallel processing algorithm especially machine learning algorithms. The main contribution of this thesis includes:

This thesis proposes a novel vision chip architecture based on hierarchical heterogeneous parallel processing (HERO). This architecture integrates image sensor, pixel-parallel, patch-parallel and distributed parallel processor, it can perform feature detection, feature extraction and feature classification algorithms and carry out various kinds of typical visual processing algorithms efficiently and flexibly.

This thesis proposed a instruction set architecture to support pixel-parallel, patch-parallel and distributed parallel programming of HERO and implemented the digital circuits include patch processing unit (PPU), processing element (PE), Lane circuit and instruction cache controller circuit.

This thesis proposed parallel feature detection, local feature descriptor algorithms include Smallest Univalue Segment Assimilating Nucleus (SUSAN) feature detector, FAST detector, SIFT detector and Local Binary Pattern (LBP) descriptor, Histogram of Orientation Gradients (HoG) descriptor. Self-Organizing Map (SOM) neural network and Adaptive Boosting (AdaBoost) feature classification algorithms are also proposed.

A HERO processor and test environment were implemented, experimental results shows that HERO can carry out the feature detection, description and classification algorithms. It achieves 64GOPS peak performance under 50MHz clock frequency, comparison shows that HERO outperforms previously reported vision chips.

Key words: vision chip; heterogeneous; parallel; feature classification; computer vision; machine learning;

目录

第 1 章 引言	1
1.1 研究背景	1
1.2 视觉芯片及其特点	3
1.3 视觉芯片的国内外研究历史及现状	5
1.4 视觉芯片设计挑战	7
1.5 论文的主要工作和贡献	8
1.6 论文组织结构	9
第 2 章 基于多级异构并行处理的视觉芯片架构	11
2.1 视觉信息处理框架	11
2.2 基于多级异构并行处理的视觉芯片架构	20
2.3 新型视觉芯片架构特点	25
2.4 本章小结	29
第 3 章 指令集以及关键电路设计	30
3.1 指令集设计	30
3.2 PPU 电路设计	34
3.3 Lane 电路设计	38
3.4 PE 电路设计	44
3.5 缓存电路设计	47
3.6 本章小节	51
第 4 章 特征检测与描述算法	52
4.1 特征点检测	52
4.2 特征描述	60
4.3 本章小节	64
第 5 章 面向视觉芯片的并行分类算法	65
5.1 图像特征识别	65
5.2 支持向量机	65
5.3 面向视觉芯片的 SOM 神经网络模型	67
5.4 面向视觉芯片的自适应提升算法	71
5.5 本章小结	73
第 6 章 多级异构并行处理器的实现与测试	75
6.1 多级异构并行处理器的实现及测试系统	75
6.2 特征点检测与特征描述算法测试	76

6.3 完整应用算法测试	79
6.4 性能比较	91
6.5 小节	93
第 7 章 总结与展望	94
7.1 本文的研究工作总结	94
7.2 工作展望	94
参考文献	97
致谢	102
个人简历、在学期间发表的学术论文与研究成果	103

第1章 引言

1.1 研究背景

人类获得信息最为直观、最为有效的方式即是通过人的视觉系统，其中我们所获得信息中超过 80% 以上均来源于我们的视觉系统。正如人类希望从鸟类的飞行中获得灵感来实现飞翔的梦想一样，我们同样希望从生物视觉系统中得到启发以构建强大的视觉信息处理系统。如图 1.1 所示，为一个简单的人类视觉系统的结构图。晶状体将入射的光聚焦投影到视网膜上，光信号在视网膜表面被转化为电信号并经过简单处理后通过视神经（Optic Nerve）被传递至视觉皮层（Visual Cortex）。视觉皮层对电信号进行分析处理，并“告诉”我们眼前所见景象是什么。然而，我们的视觉系统，尤其是视觉皮层极其复杂，直至今日人类仍然无法完全理解自身对视觉信号的处理过程。

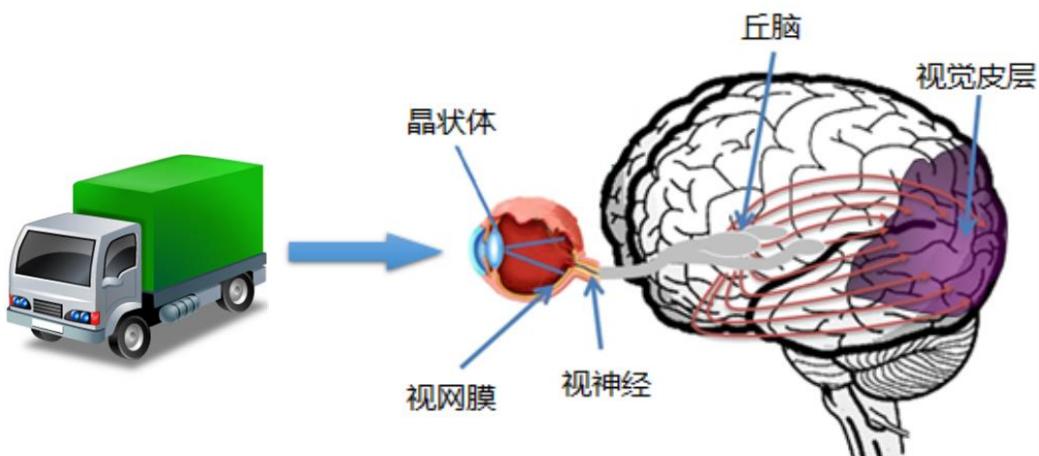


图 1.1 简单的人类视觉系统示意图

1959 年 David Hubel 和 Torsten Wiesel 两位研究者发现，视觉皮层中部分细胞（Simple Cell）对明暗以及空间位置有强烈的反应，而另一些细胞（Complex Cell）则对边缘以及运动非常敏感^[1]。他们的工作给人们呈现了视觉系统是如何将来自外界的视觉信号传递到视觉皮层，并通过一系列包括边界检测、运动检测、立体深度检测和颜色检测处理在内的过程，最

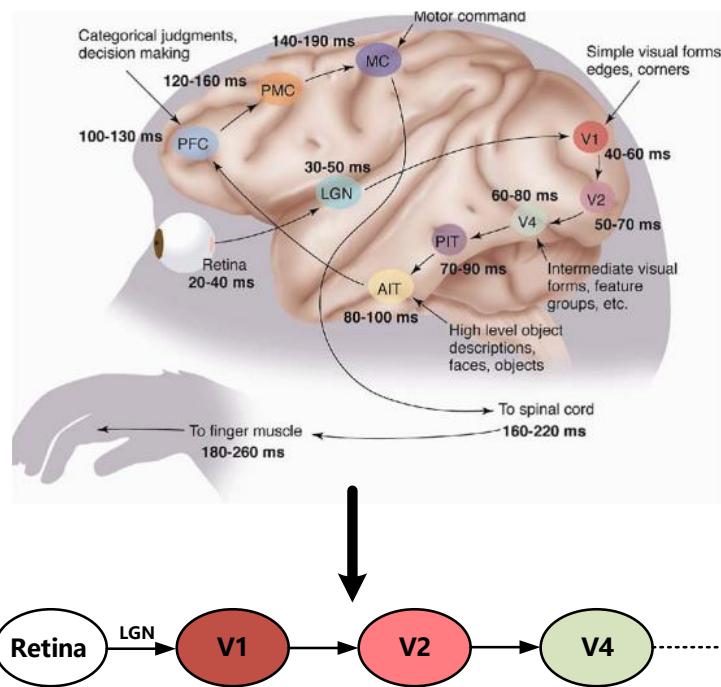


图 1.2 Gallant 和 Van Essen 所描述的层次化的视觉皮层

后在大脑中构建出一幅视觉图像的。他们卓越的工作奠定了视觉神经生理学的基础，同时也帮助他们获得了 1981 年的诺贝尔生物学或医学奖。1994 年，Gallant 与 Van Essen 两位作者发表了题为《Neural Mechanisms of Form and Motion Processing in the Primate Visual System》的论文，该文进一步揭示了灵长类动物视觉系统的神经机制。文中指出视觉皮层是一个层次化分布式系统（Distributed Hierarchy System），如图 1.2 所示，其包含了初级视觉皮层（Primary Visual Cortex, V1）以及纹外皮层（Extrastriate Cortex，例如 V2, V3, V4, V5 等）^[2]。其中低级视觉皮层 V1 主要负责对输入信号进行边缘感知，而紧接 V1 之后的次级视觉皮层（Secondary Visual Cortex）V2 和 V4 区域则与目标识别（Object Recognition）关系紧密^[3]。

到目前为止我们仅仅初步认识了生物视觉系统，其中还有很多关键问题没能解决，比如，我们是如何将简单的脉冲电信号抽象成我们所认识的事物的？我们的视觉系统是如何通过我们从婴儿时期开始对世界的观察来完善自己的？从数学角度来讲，我们的视觉系统是“运行”的什么算法？是一个统一的算法还是一连串不同的算法？尽管这样一些问题到目前为止还悬而未决，但是大量的研究工作者已经试图在数学、算法、电路以及系统层面上对某些我们已经认识的生物视觉系统功能进行建模或者实现。比如，CMOS 图像传感器，它的物理目的和视网膜类似，即将光信号转换

为电信号；加伯变换（Gabor Transform）被认为是视觉皮层中 V1 区域的数学模型^[4]；SIFT 算法这一计算机视觉领域开创性的算法，其最初灵感也是来源于对生物视觉感知的理解^[5]。这样一系列的研究极大地改变了世界以及人们的生活，CMOS 图像传感器现在被广泛的运用在数码相机、实时监控器、遥感卫星等多个领域；加伯变换、SIFT 算法则用于目标跟踪、识别、图像拼接等场合。

可以看到对生物视觉系统的研究极大的激发了各个学科研究者的创造力和想象力，而视觉芯片正是在这样的背景下孕育而生的。视觉芯片最初的设计思想即是在单一芯片上集成视觉处理系统中视觉信号获取和处理的功能，并且尽量模拟生物视觉系统结构，使其具有生物视觉系统的优点，并达到或者完成部分视觉功能。

1.2 视觉芯片及其特点

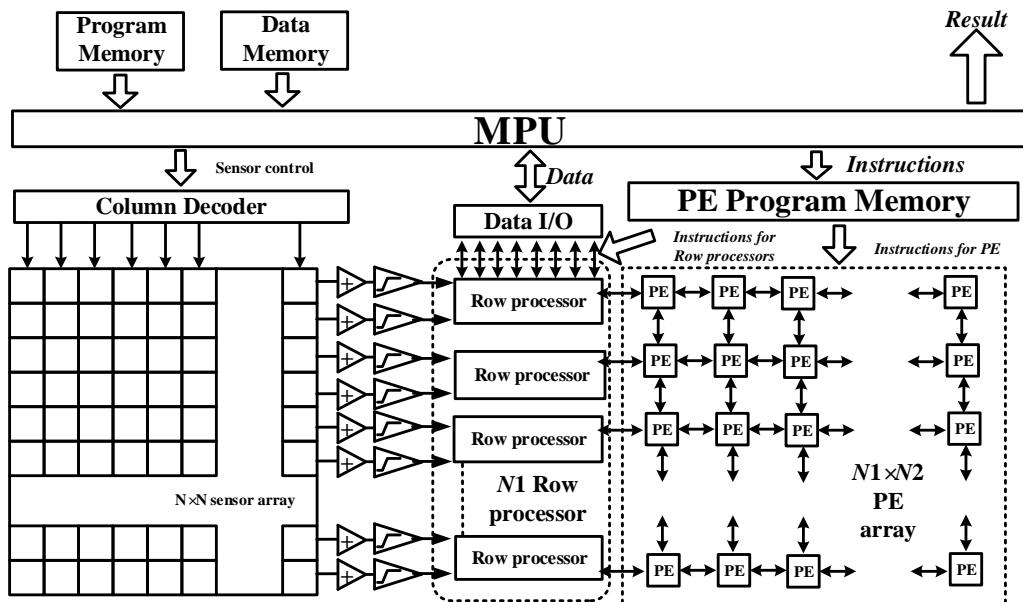


图 1.3 视觉芯片的逻辑构成，其主要组成部分为感光像素阵列和多个层次的处理器

图 1.3 所示为典型视觉芯片的逻辑构成，它集成了图像传感器的像素阵列和多个不同功能的多个处理器。在系统逻辑结构上视觉芯片非常类似于层次化异构的人类视觉系统，图像传感器类似于视网膜的成像功能，而后不同功能的处理器则类似于不同层次的视觉皮层，比如阵列处理器的功能和 V1 皮层类似，用于感知边缘以及强度信息，而后续的处理器功能则类似于次级皮层，用于对初级皮层提取的边缘信息进行整合和抽象。更为

重要的是，视觉芯片对视觉信息的处理采用并行处理，这一点与生物神经系统类似。

通过上面的简单分析，我们很容易发现，视觉芯片借鉴了很多生物系统特性，并且在现代半导体工艺快速发展的推动下，视觉芯片具有了以下特点：

- **实时以及高速特性：**视觉芯片具有并行计算和并行传输的特性，而该特性使其在计算速度上优于传统视觉处理系统。传统的视觉处理系统采用分离的图像采集部件（CMOS 或者 CCD 图像传感器）和图像处理部件（PC、DSP、MCU），其处理速度受限于图像采集和处理部件之间数据传输的速度以及并行处理部件的速度。视觉芯片的数据传输以及处理皆采用并行方式，使处理速度大大提升，很多采用视觉芯片的系统都能使响应速度达到 1000 帧/秒以上。
- **低功耗：**视觉芯片具有低功耗的特点，其采用大量并行电路并行进行数字信号处理，与传统串行处理方式相比，并行处理电路通过大量减少处理中所需的系统控制操作（循环、跳转操作，甚至是操作系统）来减低功耗。同时，片内数据传输与传统图像传感器和 PC 间数据传输相比功耗也减少很多。
- **紧凑小型系统：**视觉芯片高度集成，与传统视觉处理系统相比，其体积不足前者的百分之一。它不仅适用于传统视觉系统所擅长的一般场合，在车载、移动、物联网、机器人视觉等多种小型化平台上同样适用。
- **低成本：**视觉芯片具有低成本的特点，与传统视觉处理系统相比，几乎所有的系统部件都集成在单一硅片上，所以视觉芯片的成本要低得多。随着集成电路技术的进一步发展，相同数量晶体管的设计所消耗的硅面积还在进一步缩小，因此视觉芯片的成本还可以继续下降。

可以看到的是，视觉芯片具有传统视觉处理系统所不具备的诸多优势，如果我们能够将这些优势逐一发挥，那么视觉芯片必将在我国的国防安全、国民生产、人民生活的方方面面发挥重要作用。

1.3 视觉芯片的国内外研究历史及现状

视觉芯片的概念最早于上个世纪 90 年代末提出^[6, 7]，由于其结构和功能新颖，在短时间内即成为研究热点，并伴随出现了很多的研究小组，其中有的小组对其进行了长达数十年的研究。伴随半导体技术成熟以及各国

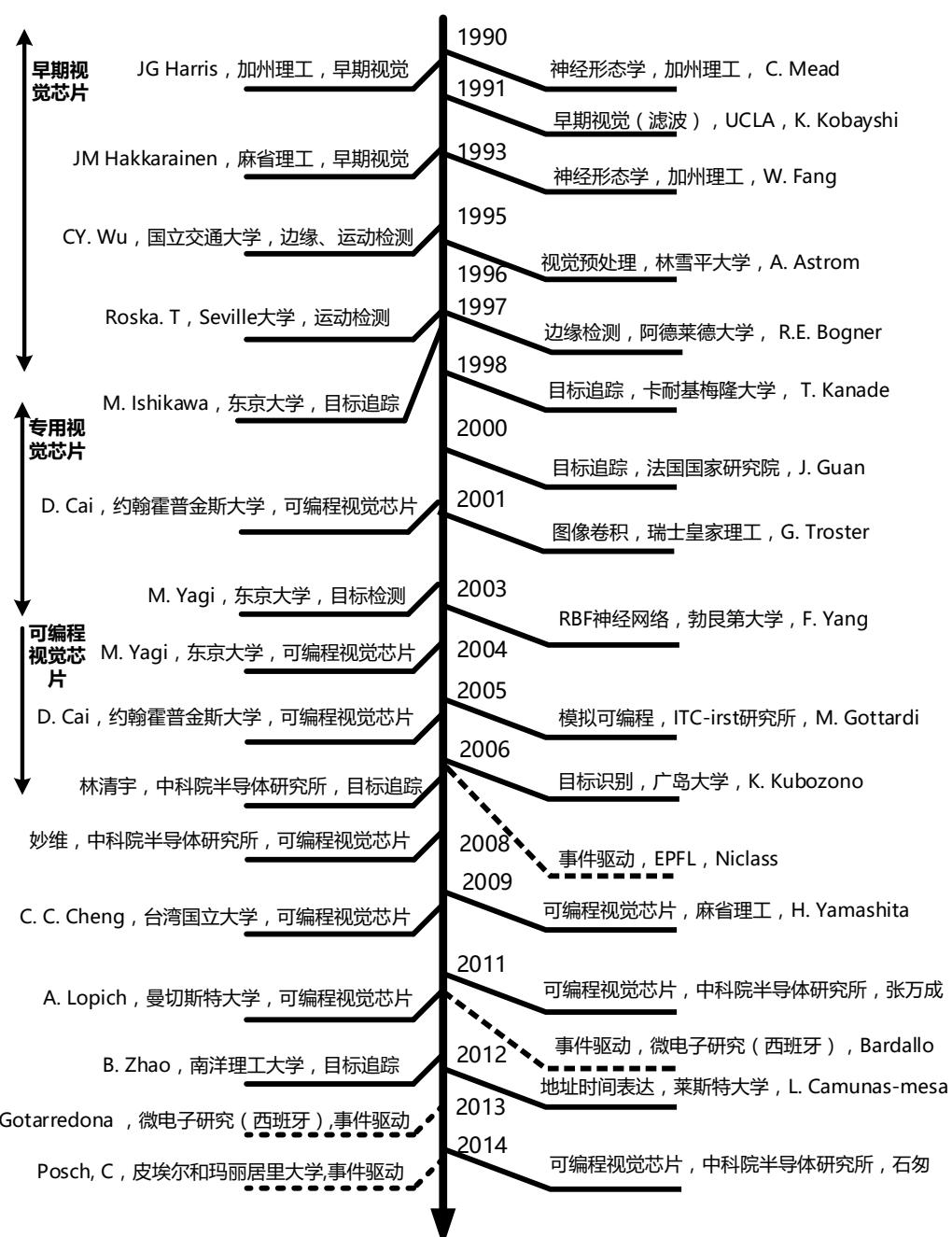


图 1.4 视觉芯片的发展

学者对视觉处理的认识加深，已形成了多个视觉芯片的研究分支。图 1.4 总结了从 1990 年开始至 2014 年为止，具有代表性的视觉芯片研究。

早期的研究者多直接利用仿生机理来设计视觉芯片，这类芯片可以将光信号转化为电信号，并且对所获得电信号进行类似视网膜功能的图像预处理。比如 Koch Christof 在 90 年代初期就利用模拟电路进行了大量的尝试，这类芯片能够在某种程度上模拟神经系统中的一些特性并且可以完成早期视觉处理（Early Vision Processing）^[8-10]。后来这类研究进一步发展为专门的神经形态学（Neuromorphic）研究，近年来如 IBM，Stanford 以及曼切斯特大学等所进行的诸多研究项目皆于此有深厚渊源^[11-13]。

90 年代初期受神经形态学启发而来的视觉芯片无法完成复杂的视觉处理，显然，仅仅进行视觉预处理并不能满足人们对视觉系统的要求。为此进入 21 世纪之后，很多学者设计了具有某些特殊功能的视觉芯片，比如用于运动检测、目标跟踪、目标判别等应用的专用（Application Specific）视觉芯片。在该方向研究工作较为突出的为东京大学^[7, 14, 15]、意大利 FBK（Fondazione Bruno Kessler）^[16-18]、新加坡国立大学^[19]以及中科院半导体研究所^[20, 21]等研究机构与大学。专用视觉芯片的优点是能够根据具体应用优化系统架构和电路结构，使芯片面积、功耗等做到最优。然而，即使是在这些固定的应用场景下，专用视觉芯片的表现可以说仍然不尽人意，究其原因，主要是芯片内所固化的算法鲁棒性不高。这一问题我们很容易从用于目标追踪的视觉芯片中看出来，这类芯片尽管采用了高速设计，但是其核心的算法却是基于二值图像进行的^[14, 15, 19, 21]。最终结果就是这类芯片都无法用于光照变化、背景复杂且灰度与目标相近的追踪任务当中。专用功能芯片的另一大局限性在于其不能应用到新的应用中，针对不同应用必须重复开发，因此其上市时间（Time To Market）相对较长，并且消耗人力物力。

2006 年以来，出现了一种新型的视觉芯片（图 1.4 中虚线所示），该类型的视觉芯片基于事件驱动（Event-driven）进行信息处理，没有帧的概念（frame-free），具有低功耗特点，并且可以在事件发生过程中进行处理，其工作原理完全不同于其他视觉芯片^[22-24]。然而，这类芯片只能对动态场景进行处理，无法感知静态图像，因此仍然只能应用于某些特殊场合。

可编程视觉芯片是为了解决专用功能芯片所存在的问题而发展而来的。它不再局限于某个固定的应用，利用其可编程性，针对不同的视觉处理，我们可以设计并在视觉芯片上运行不同的算法，这样同样一个视觉芯片就

可以在不同的场合下使用。同时，系统的鲁棒性更多的决定于视觉芯片所运行的算法，因此我们可以通过不断的提出新的算法并改进算法来提高系统的鲁棒性。因为上述优势，可编程视觉芯片在最近几年成为视觉芯片领域最为活跃的研究方向之一。美国麻省理工大学^[25]、英国曼切斯特大学^[26, 27]、国立台湾大学^[28]、美国乔治亚理工^[29]、以及中科院半导体研究所^[20, 21, 30-32]皆为较有代表性的课题组。可编程视觉芯片仍然保留了早期仿生机理以及专用视觉芯片中所采用的像素级并行处理单元（Processing Element, PE）阵列以保证整个系统的处理速度。近期出现的一些研究成果中使用了行/列处理器（Row Processor, RP）阵列和微处理器（Micro Processing Unit, MPU）相结合的方式来进一步完善图像特征提取和识别功能。

1.4 视觉芯片设计挑战

如上节所述，视觉芯片领域已经取得了一系列研究成果。然而，进入 21 世纪以后，尤其是 2010 年之后，人类在计算机视觉、机器学习、人工智能等领域的认识和技术发生了飞跃性、爆发式的发展，这些技术不仅极大的改变了世界的格局和人类的生活，同时也给视觉芯片的研究注入了新活力、提供了新方向。以计算机视觉为例，目前人脸检测、人脸识别、目标追踪等最为常用的视觉应用算法皆出于此，但是目前所报道的视觉芯片中还没有一款可以完整的实现这类算法。利用最先进的机器学习方法得到的神经网络对图像的识别与分类能力甚至已经超过了人类^[33]，这类算法更好的对人类层次化的视觉系统进行了模拟，然而视觉芯片仅仅在人工神经网络方面有了初步尝试^[31]。因此，本文认为，视觉芯片的研究应该和国际国内相关领域的发展并行开展，具体的应该从以下几个方面入手：

(1) 架构上，设计能够支持先进计算机视觉算法的视觉芯片架构。目前大部分视觉芯片缺乏对先进计算机视觉算法的支持，难以完整的完成复杂的图像特征提取以及最后的特征分类过程。目前，仅有几款最近几年在 JSSC (Journal of Solid-State Circuits, IEEE) 上报告的视觉芯片初步具备这些功能^[28, 30, 32]。国立台湾大学的 L. Chen 集成特征处理器（Feature Processor）来进行图像特征提取，并使用 MIPS 处理器进行特征分类。中科院半导体所张万成则采用行处理器来提取特征，同时使用微处理器进行特征分类。虽然两者具体实现方法上有所差异，但是皆是为了解决视觉芯片中特征提取、特征分类的问题。半导体所石勿等人在 2014 年的 ISSCC

(International Solid-State Circuit Conference) 会议上报道了使用自组织映射 (Self-Organizing Map, SOM) 神经网络来并行完成特征分类的视觉芯片，虽然该工作在解决分类问题上有所贡献，但是在特征提取方面稍显不足。因此，本文认为，我们需要提出一种可灵活编程且能够很好应对视觉处理各个阶段算法的架构。

(2) 电路上，设计功能灵活、结构紧凑的处理电路。视觉芯片的电路设计应具有可编程性和易于编程性，一方面可以利用单一电路的可编程性兼顾更多的图像处理运算并减少系统面积，另一方面，设计相应电路指令集以降低算法开发难度。同时，处理电路设计要考虑其最终性能，相同的架构下，不同的电路实现会导致较大的性能差异。例如，采用流水线化设计的电路可以通过指令级并行来提高电路性能数倍。

(3) 算法上，提出新算法并对现有的视觉芯片算法进行创新与改进，同时应将目前先进的计算机视觉算法并行化、精简化并应用于视觉芯片。一些传统的视觉芯片算法在速度上有明显的优势，但是算法所能应用的范围有限且鲁棒性不高，我们应该从算法本身入手来提高鲁棒性。对于已有的计算机视觉算法，我们应该对其进行深入理解，充分的发掘其中的并行性，并通过一些简化手段，使其可以应用到视觉芯片上来。

1.5 论文的主要工作和贡献

本文的主要工作围绕可编程视觉芯片的架构、设计、算法等几个方面展开，研究基础主要建立于课题组在该领域长达十年的学术积累之上。本论文的主要贡献如下：

(1) 提出了适用于计算机视觉算法处理的基于多级异构并行处理的视觉芯片架构—HERO。该架构集成多级异构并行处理器，有效地提高了特征描述和特征分类处理性能。该架构可以完成从特征检测、特征提取到特征分类的完整视觉处理流程。

(2) 设计了在 HERO 架构上使用的指令集，该指令集加强了系统的可编程性，开发了块处理电路并设计了可完成分布式并行处理和像素级并行处理的电路，大大减少了实现像素级并行处理阵列所消耗的硬件资源。

(3) 提出了用于视觉芯片的多种并行算法，包括最小核值相似区 (Smallest Univalue Segment Assimilating Nucleus, SUSAN)，快速特征点 (FAST) 检测、LBP (Local Binary Pattern) 纹理描述子、HoG (Histogram of Orientated

Gradient) 描述子、人脸检测、基于 LBP 描述的快速目标追踪以及机器学习 (Adaptive Boosting、SOM) 算法的大规模并行实现。

(4) 实现了 HERO 架构，并搭建测试平台，开发了专用的汇编工具以及相关软件包，在 HERO 架构上成功完成了所提出各种并行算法并与其它工作进行了性能对比，整体系统性能超过目前世界先进水平。

1.6 论文组织结构

本章简单介绍了视觉芯片的研究背景及其特点，从视觉芯片研究发展角度出发对各个研究方向进行了阐述，最后结合视觉芯片的发展现况和当前相关领域的进展指出了进一步研究视觉芯片所面临的挑战并总结了本论文工作的主要贡献。

论文第二章，将从视觉信息处理框架入手介绍分析传统与现代处理方法的不同，并介绍现代视觉信息处理中的几个必要步骤及其意义，并从硬件实现的角度对算法并行度、复杂度进行分析，进而提出一种适用于现代计算机视觉算法的视觉芯片架构—HERO。该架构包括 PE 阵列处理器、块处理器 (Patch Processing Unit, PPU)、车道阵列 (Lane Array)，它们分别可以完成像素级并行、块并行以及分布式并行处理。HERO 架构能够实现特征检测、特征提取以及特征识别的完整视觉流程。

论文第三章将介绍 HERO 架构的指令集设计以及相应的电路设计。其中指令集借鉴了 MIPS 处理器指令集并在此基础上增加了大量用于并行处理的专用指令。同时，第三章还将介绍块处理器电路、Lane 电路、PE 电路以及缓存电路的设计以及这些电路完成并行处理的方法。

论文第四章提出了可用于视觉芯片的大规模并行处理的特征点检测算法和特征描述算法。其中，包括最小核值相似区特征点检测、快速特征点检测、尺度不变特征变换特征点检测、LBP 纹理特征、HoG 特征。

论文第五章提出了用于 HERO 架构的快速分类算法，介绍了支持向量机、SOM 自组织映射神经网络和 AdaBoost 算法的数学原理以及它们在 HERO 架构上的并行实现的可能性和方法。

论文第六章介绍 HERO 架构的实现以及其测试平台，并对第四、五章所提算法进行测试，同时完成了人脸检测、识别、目标追踪等应用算法。测试性能和一系列国际成果进行了对比并给出了比较结果。

论文第七章对本论文进行总结，概括本研究取得的成果，同时结合当前的研究状况和研究热点指出视觉芯片未来研究的发展方向。

第2章 基于多级异构并行处理的视觉芯片架构

本章将从视觉信息处理框架入手对传统视觉处理方法和现代视觉处理方法的优劣进行分析，并从硬件实现的角度分析现代视觉处理算法的并行度和复杂度，从而提出适用于现代计算机视觉算法的基于多级异构并行处理的视觉芯片架构，它具备像素级、块级以及分布式并行处理能力，并且兼容传统视觉芯片的各种处理算法，可高效、灵活地完成各类典型视觉信息处理算法。

2.1 视觉信息处理框架

视觉系统需要解决的核心问题是识别，其涉及到对物体以及场景的分类和识别。传统的视觉处理使用如图 2.1 所示的流程来完成图像识别，该流程由低级、中级和高级三个部分组成。在低级处理中，使用图像滤波、阈值分割、形态学操作等算法对图像进行处理以提取出轮廓、边缘以及结构等信息。中级处理在整个图像的基础上进行全局特征提取，比如在空域进行灰度直方图统计、主成分分析(Principal Component Analysis, PCA)，线性判别分析(Linear Discriminant Analysis, LDA)，在频域进行傅里叶变换、小波变换等。高级处理对中级处理提取出的特征进行识别和分类，常用的为人工神经网络、贝叶斯分类器、决策树、自适应提升算法(Adaptive Boosting, AdaBoost)等。

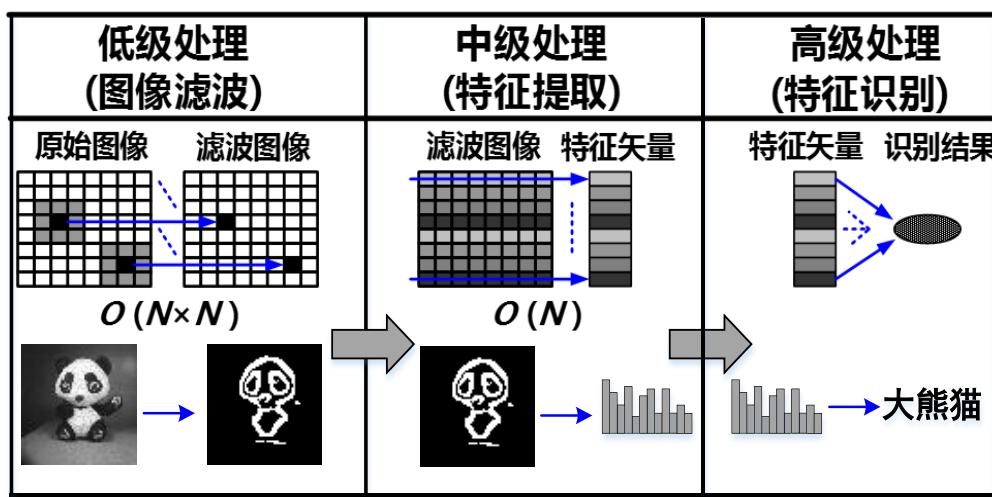


图 2.1 传统视觉处理流程

图 2.1 所示的框架在视觉处理系统中有着广泛应用，比如早期的特征脸(Eigenface)^[34]、基于主要边缘投影(Projected Principal-Edge Distribution, PPED)的人脸检测^[35]、基于傅里叶变换的图像识别、目标追踪等方法都是

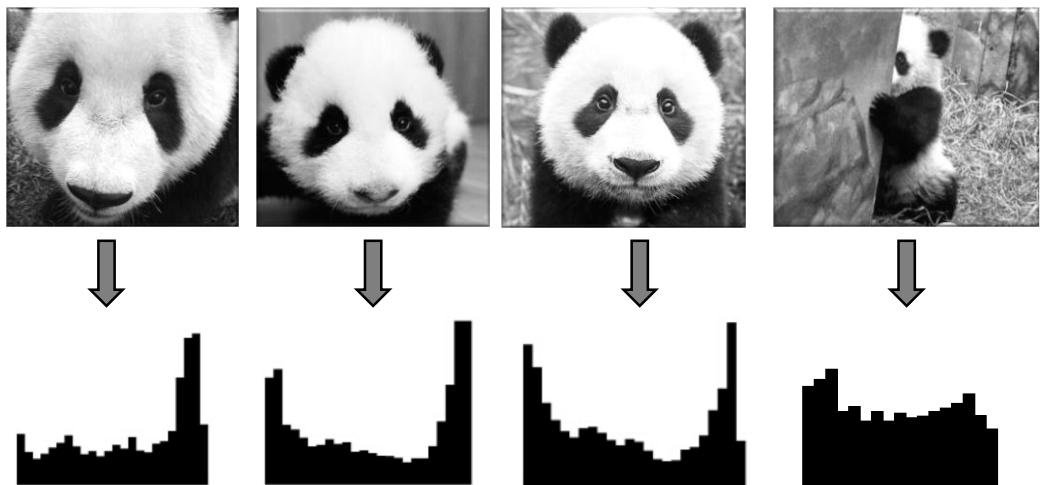


图 2.2 利用灰度直方图对图像进行表达

在上述框架内提出的。石勿等人在该框架上设计的视觉芯片^[31]实现了人脸检测、手势识别等应用算法。然而，该框架也存在着明显的不足，因为即使相同的物体在光照、角度、距离以及背景变化的情况下也会产生相差很大的图像特征，基于上述框架的全局特征很难有效的对图像进行描述，以至于图像识别任务难以完成。图 2.2 所示，为使用图 2.1 中框架方法对图像进行描述的结果，左二与左三图像比较接近，因此所得到的直方图特征也比较接近，左一在成像角度上稍有区别即导致直方图特征产生了变化。左四图中，目标被部分遮挡，背景较为复杂，所得到的直方图特征已经完全与前三者不同。我们可以看到图像特征表达阶段的不足，将导致高级处理阶段分类准确性降低，最终影响到整个视觉处理系统的性能。

传统的视觉处理框架是无法与生物视觉系统比拟的。以检测人脸为例，人类在任意视角、剧烈光照变化、甚至强遮挡的情况下只要能看到人脸的局部，比如眼睛、鼻子、嘴巴、甚至只是皮肤或者轮廓即可准确的判断出眼前所见是否为人脸，《琵琶行》中所描述的“犹抱琵琶半遮面”，就可以看作一个这样的例子。那么我们能否在算法上实现生物视觉系统所采用的这种“局部”特征来实现高效准确的识别呢？

不列颠哥伦比亚大学（University of British Columbia）的 David G. Lowe 教授在 2004 年提出的尺度不变特征转换 (Scale-invariant feature transform, SIFT) 算法^[5]是第一种成功利用“局部”特征来进行图像识别的算法。该算法首先筛选出图像中值得进行描述的局部区域，然后对这些区域进行局部特征描述，最后再利用这些局部特征进行图像识别。SIFT 算法是最为成功的局部特征表达算法之一，它的提出对计算机视觉做出了巨大贡献，并且在图像拼接、识别、目标追踪等应用领域发挥了重要作用。现代视觉处理框架正是使用了局部特征描述的方法来解

决传统视觉处理框架在光照、畸变、遮挡下鲁棒性差的问题。如图 2.3 所示，在该框架中，首先利用特征检测，可以得到图像中具有显著特征的区域，亦或直接将图像划分为多个均匀的图像块，然后使用特征描述算法对这些区域进行局部特征描述。这与传统的视觉处理流程有显著的不同，所有特征均来源于图像的局部区域，而非整个图像。这些特征被称为局部特征，每个图像中可以获得数量可观的局部特征，与传统视觉处理相比，这种方法所获的特征数量更加丰富，更加接近生物视觉系统的特点，因而更加适合于图像识别。利用特征检测和特征描述结合的方法即得到图像的局部特征表达，这些局部特征构成的向量最后通过机器学习算法进行分类，即可实现检测、追踪、识别等视觉处理能力。

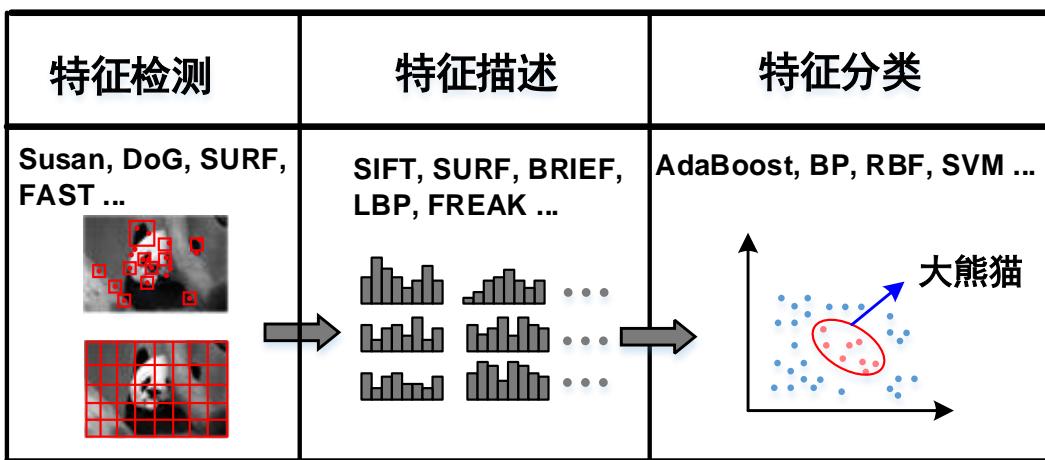


图 2.3 计算机视觉处理流程

本文将根据图 2.3 的视觉处理框架设计新型视觉芯片架构，为此在后续几个小节中我们将首先对特征检测、特征描述以及特征分类进行介绍，然后分析其各个阶段的并行度。

2.1.1 特征点检测

特征检测的目的是提取出图像中值得描述的区域，这些区域应该在图像进行变换之后，也就是在图像噪声、不同视角以及不同照度下仍然具有显著

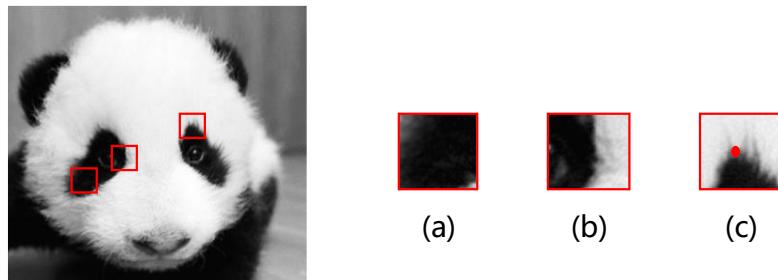


图 2.4 特征点示意图

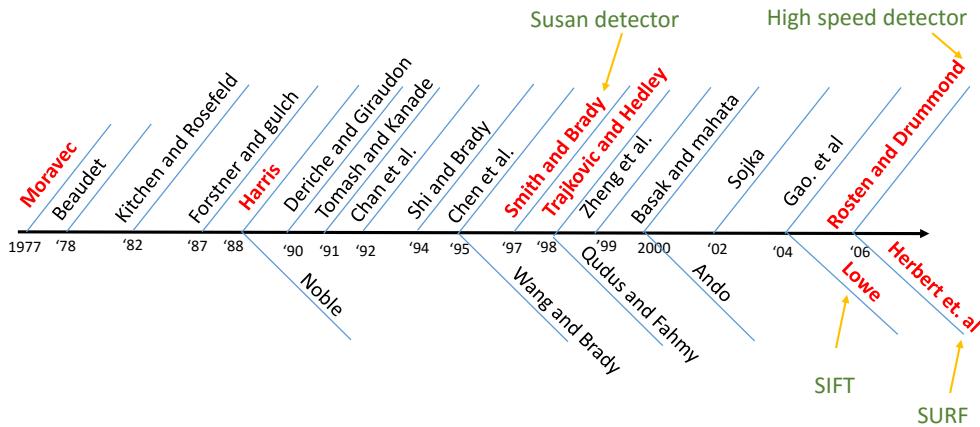


图 2.5 角点检测算法的发展

特征的区域，这些区域的特点是灰度在两个或者更多方向变化。如图 2.4(a) 为一个均匀灰度的区域，这样的区域不具备特异性，无法提取出有用的特征信息，图 2.4(b) 虽然具有明显的边缘，但是灰度只在一个方向上变化，如果观察视角发生变化，同样难以辨别。图 2.4(c) 中，灰度在两个方向都有剧烈的变化，这样的区域，在图像旋转、角度变换等情况下仍然容易被辨别，最适合作为特征区域。在两个方向的灰度剧烈变化便会产生图 2.4(c) 中的特征点，因此通过特征点检测就可以确定需要进行特征描述的区域。特征点也被称为角点，本文后面的内容将对两者不加区别的使用。事实上，特征点检测算法的由来已久，图 2.5 反映了角点检测算法的发展历史。最早的角度检测算法由 Moravec 在 70 年代末提出^[36]，在 80 年代末，Harris 和 Stephens 又在此基础上提出了更具鲁棒性的 Harris 角点^[37]，1997 年 Smith 和 Brady 提出了 SUSAN 角点^[38]。然而，早期的角度检测算法对高性能角点（在不同照度、旋转、变换等情况下可以被重复检测的角点）的检测鲁棒性较差，直到 06 年左右 SIFT^[5]、SURF^[39]、FAST^[40]等一系列算法的提出才解决了提取高性能角点的问题。在第四章中，我们将详细介绍特征检测算法，并提出其并行实现方式。

2.1.2 局部特征描述

在利用特征检测算法得到对图像中显著的局部区域后，便可以使用局部特征描述对这些区域进行描述。仍然以人脸的检测为例，不难想象经过角点检测，在眼角、鼻尖、嘴角等区域都会出现大量的特征点。那么对这些特征点所在局部区域的描述也就是对眼睛、鼻子、嘴的描述，也就达到了在“犹抱琵琶半遮面”的情况下仍然能够通过对眼睛、鼻子等脸部特征的

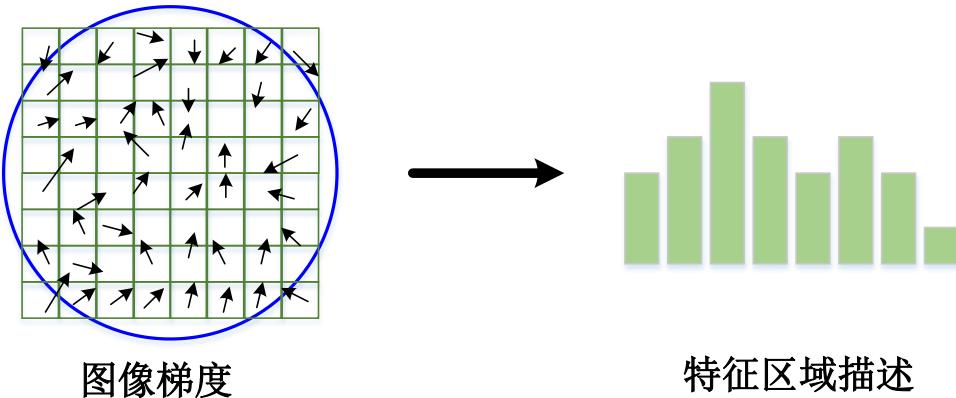


图 2.6 SIFT 特征描述

识别来达到检测人脸的目的。这里所指的描述，主要是针对图像局部区域的某种或者几种特征的统计描述，通常采用构建直方图的方法。图 2.6 为 SIFT 算法中进行特征描述的示意图。蓝色圆圈所在区域即是角点所在局部区域，对该区域内每个像素点梯度方向和强度进行统计所得到的直方图就是 SIFT 局部特征。不同的算法统计的特征有所不同，比如局部二进制纹理描述（Local Binary Pattern, LBP）算法^[41]，就是对局部纹理特征的统计，该算法适用于目标纹理丰富的图像识别中。方向梯度直方图（Histogram of Oriented Gradient, HOG）算法^[42]则是对局部区域中像素梯度方向进行统计而得到的，该算法广泛应用在行人识别、车辆识别等梯度方向明显的图像识别中。在第四章中，我们将详细介绍这些描述子以及它们如何并行实现。

在 Lowe 之后十几年，尽管用于特征识别、匹配的计算机视觉算法层出不穷^[7]，但都秉承了 SIFT 算法局部特征描述的方法，并且这一方法到目前为止仍然是解决图像识别问题的重要手段之一。当然，值得注意的是，最近几年出现的很多算法省略了特征点检测的步骤，比如 LBP 和 HOG 算法直接在对整个图像不加区别的进行局部特征描述，同样的取得了非常好的效果^[42, 43]。目前，深度学习（Deep Learning）的提出和发展使计算机视觉领域发生了深刻的变化，使用深度学习方法，我们不再需要人工的设计图像特征而是通过深度神经网络自主提取图像特征^[44]。这一过程虽然和上述过程看上去稍有差异，并且往往也不再使用直方图作为特征的描述手段，但是其本质上仍然可以认为是对图像的局部特征描述。总而言之，我们需要对图像采用了局部描述的方式才能进行更加有效的视觉处理。

2.1.3 特征分类

当进行完局部特征描述之后，就可以对特征进行分类了。分类的过程也就是模式识别（Pattern Recognition）的过程，主要使用到一些机器学习算

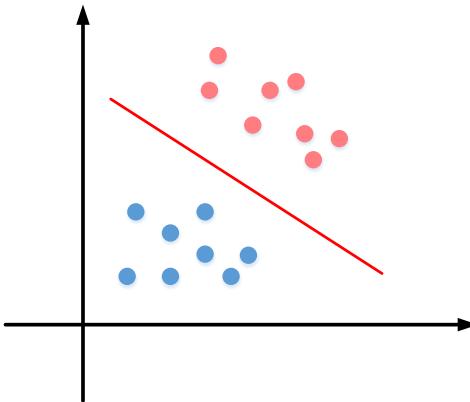


图 2.7 一个将二维平面上两种类型的点进行分类的例子

法，如 AdaBoost^[45]，支持向量机（Support Vector Machine，SVM）^[46]以及人工神经网络（Artificial Neural Network，ANN）^[47]。这类算法通过对大量样本（Labeled Sample）进行复杂的训练来提取出样本集的数学模型（Hypothesis）并以此来判断新的数据特征应该属于哪一类。

以一个二维平面中的数据分类为例，如图 2.7 所示，平面上有两种不同的点，分别用不同的颜色来表示，一种为红颜色，而另一种为蓝颜色。很显然，如果我们要区分这两类点的话，一种分类的方法就是找到一条在该二维平面上分割这两类点的直线。显然，图中的红线即是一条可行的分类直线，如果我们令该直线的方程为：

$$f(x) = w^T x + b \quad (2-1)$$

显然，直线上的点应满足 $f(x) = 0$ ，蓝色的数据点满足 $f(x) < 0$ ，而红色的数据点则满足 $f(x) > 0$ 。这个直线方程就可以看作我们对该两类数据分类的数学模型。很多分类算法的目的就是在数据集中找到上述分割数据的“直线”，对于图像这样的高维度数据而言，对数据进行分割的是超平面。

关于 AdaBoost、SVM、人工神经网络的具体算法流程，本文第 5 章将着重介绍。在这里，我们简单罗列这几个算法的最后数学形式，以便在下一小节对其并行度、复杂度进行分析。式 (2-2) 为 AdaBoost 算法的分类函数表达式，其中 a_i 为弱分类器 $h_i(x)$ 的权重因子（Weighting）， T 为一个常数，弱分类器 $h_i(x)$ 输出为 0 或者 1。式 (2-3) 为支持向量机的分类函数表达式，其中 x_i 为支持向量， a_i 和 y_i 分别为支持向量 x_i 所对应的权重和类别。式 (2-4) 为 SOM 神经网络的分类函数， FV 为需要分类的特征向量， $RV_{i,j}$ 为神经元 (i,j) 中的参考向量， FV 和 $RV_{i,j}$ 皆为 K 维度的向量。与特征

向量距离最近的参考向量所在神经元的位置即代表了该特征向量的类别。

$$C(x) = \text{sign}(\sum_{t=1}^T a_t h_t(x) - T) \quad (2-2)$$

$$f(z) = \text{sign}(\sum_{i=1}^{totalSV} y_i a_i \langle x_i, z \rangle - \rho) \quad (2-3)$$

$$Dist_{i,j} = \operatorname{argmin} \| FV - RV_{i,j} \| \quad (2-4)$$

2.1.4 视觉处理并行度分析

以往视觉芯片在设计时考虑图 2.1 中的视觉处理框架，认为对于 $N \times N$ 大小的图像，初级和中级阶段分别具备 $O(N \times N)$ 像素级并行度和 $O(N)$ 行并行度^[48]，并以此为依据进行架构设计。这些架构可以高效的执行传统视觉处理算法，比如完成中低级滤波、地平线检测、手势识别等应用。然而，因为设计考虑不同，基于这种设计思想的架构无法很好的适应现代视觉处理算法。经过大量的分析，我们认为，现代视觉处理中，主要涉及像素级并行、块级并行以及分布式并行三种并行度，新型架构应该对这三类并行度提供良好的支持以实现高效的视觉处理。下面我们首先对着三类并行度进行分析与介绍。

(1) 凡是利用邻域像素值进行计算即可直接得到每个像素结果的算法便可以采用像素级并行处理。SIFT 角点检测中使用的高斯滤波、FAST、SURF 角点检测算法使用的邻域像素灰度值比较等都是利用邻域像素值进行计算的，它们和图像滤波、背景减除、阈值分割、数字形态学以及部分特征生成的算法一样，每一个像素的处理仅和其邻近范围内的少数像素有关。这样一些算法，我们可以用图 2.8 中的串行程序来统一表示。该程序中两个嵌套的 *for* 循环用于对原始图像像素值进行扫描操作，*width* 和 *height* 分别为图像的宽和高，*func* 为简单的多项表达式，用于计算每个像素的结果。比如，在大部分的算法中，上面代码中的 *out_image[x][y]* 就只与 *px[x][y]* 周围 3×3 或者 5×5 范围内的像素有关，并且只涉及到简单的加、减运算，对于形态学运算，则只需与周围的像素进行简单逻辑运算。

从串行程序实现可以看出，*out_image[x][y]* 的计算并不依赖于任何之前的运算结果，因此，在硬件资源允许的情况下，所有 *out_image[x][y]* 的计算完全可以并行的进行，也就是如图 2.8 中像素级并行代码，消除了两层 *for* 循环，大大提高了算法的性能。

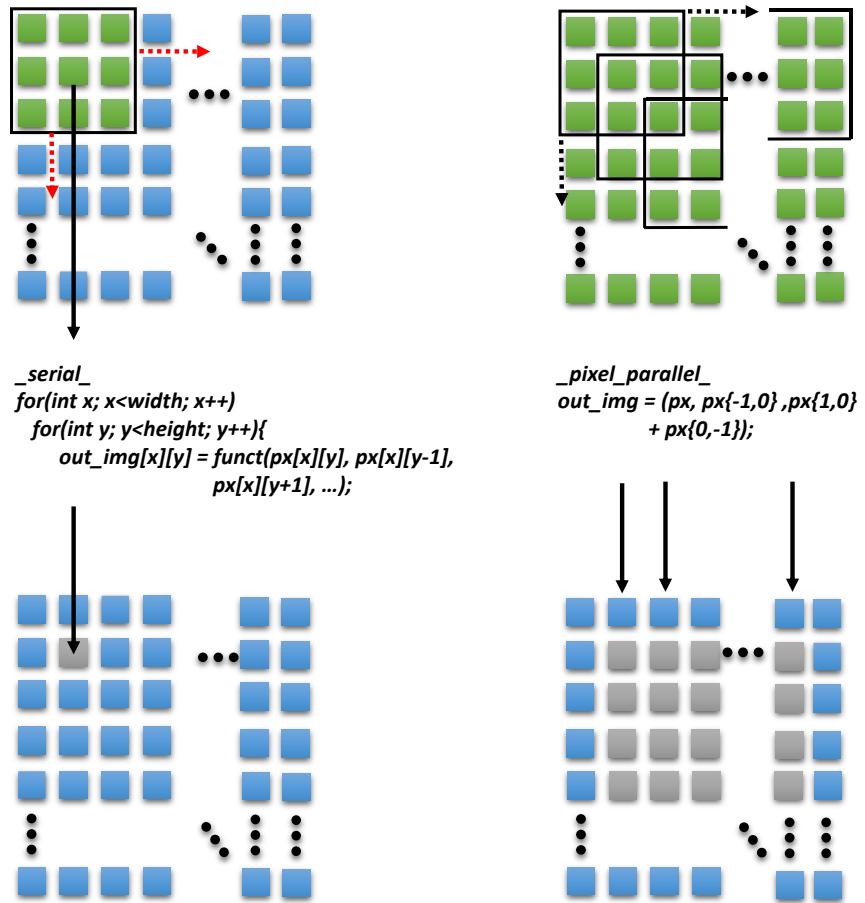


图 2.8 视觉算法的像素级并行处理

(2) 在视觉处理中，很多操作是基于图像块进行的。例如，LBP 纹理描述时，需要对一块局部图像的纹理特征进行统计，HoG 则对小块图像中像素的梯度方向进行统计，SIFT 算法中特征提取也是在一小块特征区域中进行的，这样的特征统计操作可以统一用图 2.9 中的 C 语言串行代码表示。其中，*width* 和 *height* 分别为图像局部图像块的宽和高，*patch_num* 为需要处理的局部图像块数目。对于第 *n* 个图像块，其统计特征保存在 *feat_vect[n]* 数组中。显然，该代码是无法像素级并行执行的，因为 *feat_vect[n]* 数组中任意一个元素的结果都有可能依赖于里面两层 *for* 循环的计算结果。但是，二维数组 *feat_vect* 中各个元素之间没有相关性，因此是可以同时计算的，也就局部图像块之间的计算可以并行进行。局部特征描述可以并行处理，我们称这种并行为块并行处理。

图 2.9 中呈现了块并行执行的代码，虽然最外层循环被并行执行，内层的两个 *for* 循环仍然需要进行计算，因此块并行时需要灵活的可编程电路才能实现，比如支持寄存器寻址，程序跳转等复杂的程序控制操作的电路。

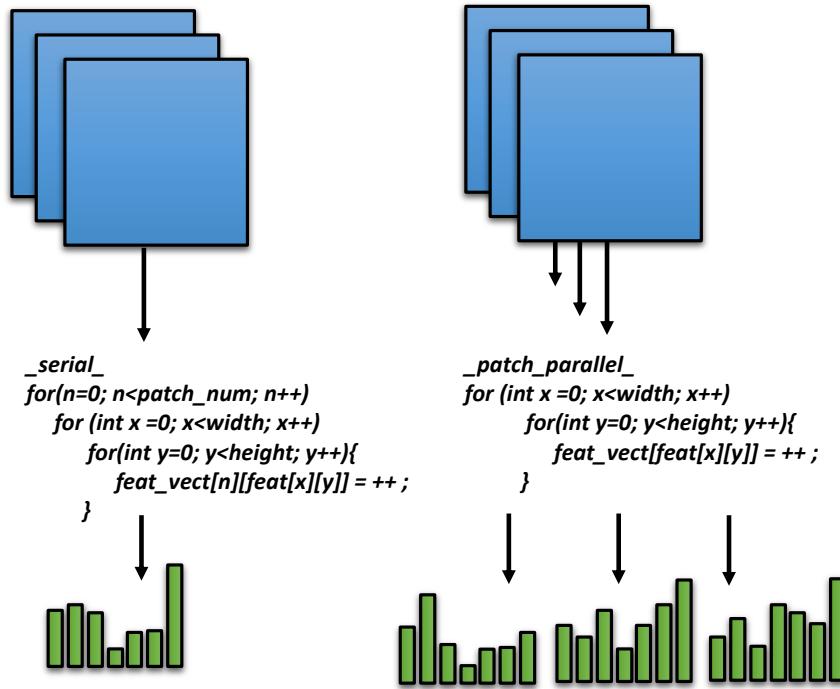


图 2.9 视觉算法的块级并行处理

在第三章中，我们可以看到图 2.9 中块并行处理代码的两个 *for* 循环虽然无法完全并行，但是也可以通过并行电路实现一定程度的加速。

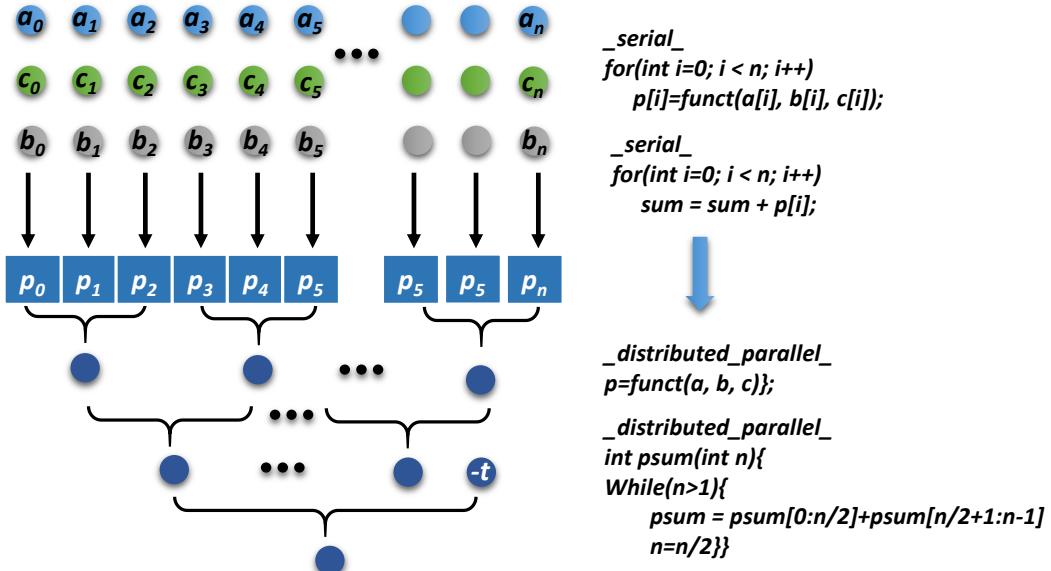


图 2.10 分布式并行处理的 C 语言代码

(3) 用于图像识别的特征维度很大，比如 AdaBoost 由数百个弱分类器构成、SOM 神经网络中包含数百个神经元，支持向量机中支持向量更是有可能多达千个以上。如果采用串行计算，这部分工作往往成为整个系统的瓶颈^[28, 31, 32]。为提高系统性能，特征分类同样需要并行处理，那么这些

算法怎么并行呢？通过对几种分类算法的研究，本文认为，分布式并行算法大多可以使用图 2.10 所示的 C 语言串行代码表示，其中 a , b , c 和 p 皆为数组， $funct$ 为对 a , b , c 数组中元素进行内积或者乘法运算的函数， p 数组中每一个元素保存经过计算得到的子项。比如在式（2-3）、（2-3）以及（2-4）中大量的子项计算就可以用相应的 $funct$ 计算得到。在实际处理时，我们可以采用分布式并行的方式来对分类算法进行加速，因为各个子项之间不存在相关性，因此可以像图 2.10 中的并行代码一样将第一个 *for* 循环并行执行。最后对这些子项采用两两合并求和即可快速得到最后的分类结果。

2.2 基于多级异构并行处理的视觉芯片架构

基于上一节对现代计算机视觉算法特点的分析，本文提出了基于多级异构并行处理（Hierarchical hEterogeneous parallel pROcessing, HERO）的视觉芯片架构。如图 2.11 所示，该架构包括图像传感器，系统总线，传感器控制模块，传感器缓存、程序缓存、网络接口，4 个用于大规模并行处理的计算核心（Computation Core, CC）以及一个 MPU 微处理器。传感器缓存用于存储来自图像传感器的数据，以便于处理器多次访问。程序缓存中存储了用于视觉处理的算法程序。网络接口用于与 PC 通信。4 个计算核心为 HERO 架构提供了主要的计算性能并且可以实现多指令多数据（Multiple Instruction Multiple Data, MIMD）工作。根据 MPU 的控制指令，四个计算核心之间可以进行相同或者完全不同的操作。

每个计算核心的结构如图 2.12 如所示，由一个主要处理器（Main Processor, MP）和三个虚拟处理器（Virtual Processor, VP）组成。它们都由块处理单元（Patch Processing Unit, PPU）、32 个车道组成的车道阵列（Lane Array, LA）以及 16×32 个 PE 组成的 PE 子阵列（Processing Element Sub Array, PESA）。如图中所示，每个 Lane 电路中包含了 16 个 PE 处理单元，32 个 Lane 中所有 PE 共同组成了 16×32 的子 PE。两个相邻的 PE 子阵列构成了 32×32 的 PE 阵列。MP 中的 PPU 不仅包含了解码器、寄存器组、逻辑运算单元（ALU）、数据存储器（MEM）还集成了一个指令缓存和程序计数器（Program Counter, PC）用于取值和程序计数。同一个计算核心中只有 MP 中的 PPU 具有取指令能力，VP 中的 PPU 执行和它相同的程序。由于 PPU 具有独立的寄存器组，因此虽然它们执行的指令一样，但是所操作的数据却完全不同。

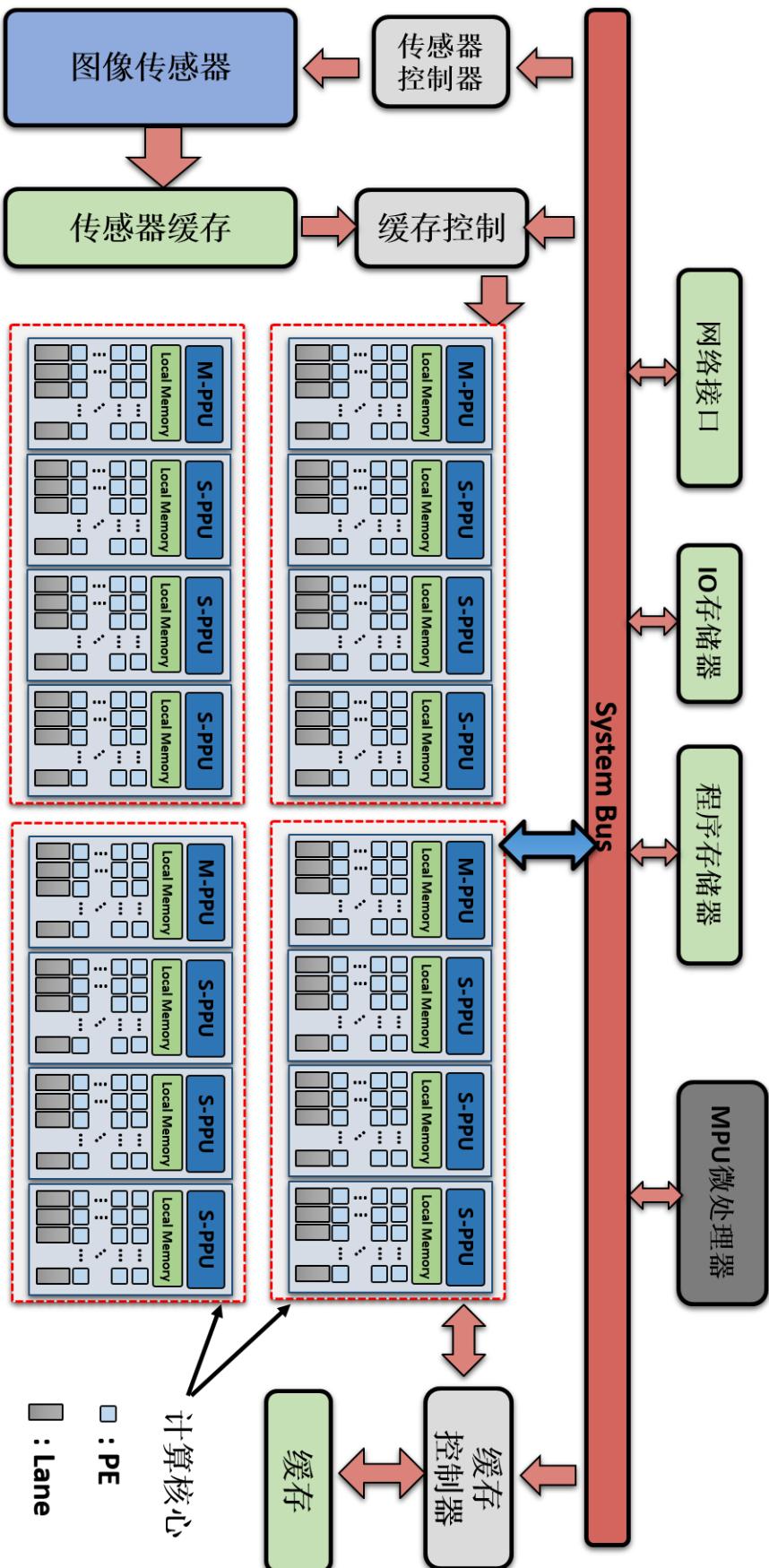


图 2.11 HERO 架构

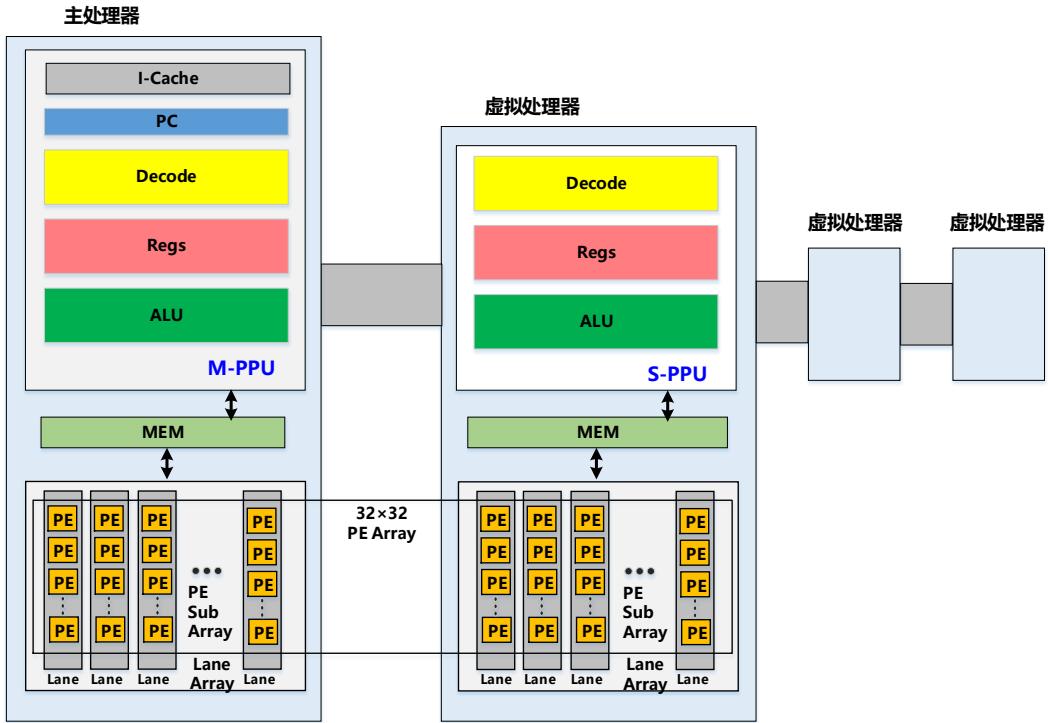


图 2.12 HERO 中计算核心示意图

图 2.13 为 HERO 中主处理器示意图, 其中 PPU 采用了五级流水的 MIPS 处理器架构, 包含取值 (Instruction Fetch, IF)、指令解码 (Instruction Decode, ID)、执行 (Execution, EXE)、存储器周期 (Memory, Mem) 以及写回 (Write Back, WB) 五个流水级。PPU 支持大部分 MIPS 指令。车道阵列由 32 个一维局域相连的 Lane 组成, 其执行的指令由 PPU 指令解

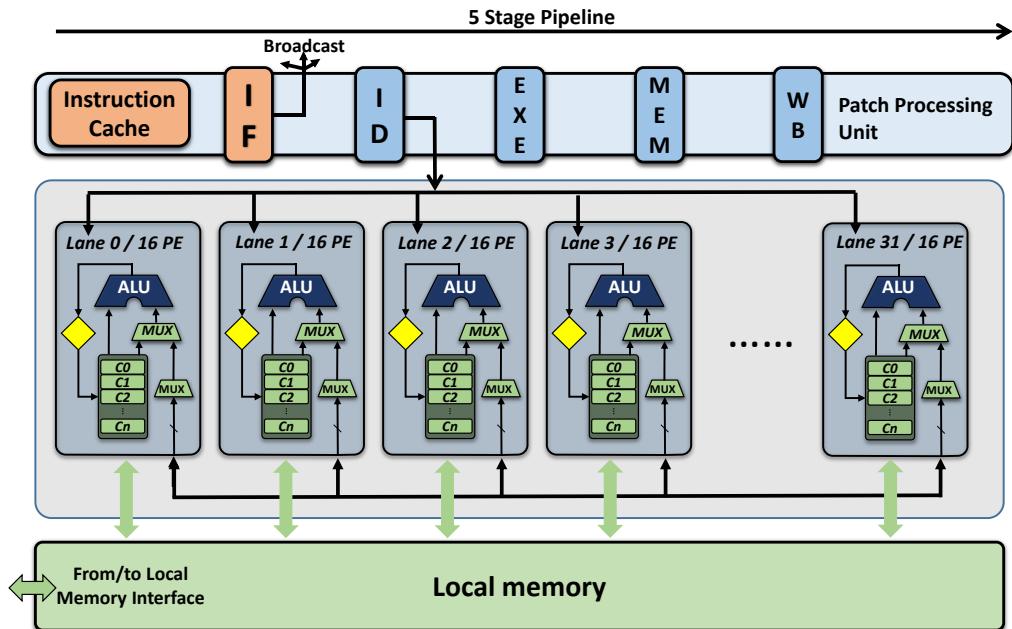


图 2.13 HERO 中主处理器示意图

码获得。在第三章中，我们将具体分析每个 Lane 是如何等效为 16 个 PE 单元以及 Lane 阵列是如何等效为 16×32 的 PE 阵列的。PE 阵列可以完成图像滤波、形态学、特征生成等算法并且可以访问 PPU 的数据存储器。PE 阵列所执行的指令也是由 PPU 指令解码后获得的。

如图 2.14 所示，为 HERO 架构可完成的不同粒度的处理，其中包括像素级并行处理，块并行处理，以及分布式并行处理。通过时分复用，像素级并行处理和分布式并行处理使用相同的硬件来完成，PPU 处理器主要用于完成块级并行处理中所需的大量程序控制以及复杂计算。PE，Lane 和 PPU 之间共享存储空间，MPU 主要用于系统管理。

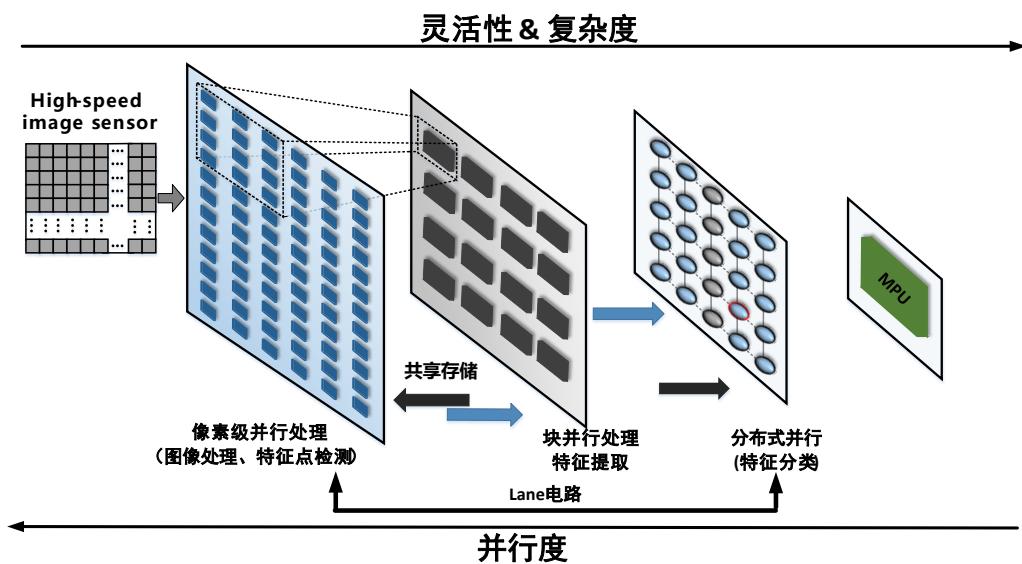


图 2.14 HERO 完成不同粒度视觉处理的示意图

与先前报道的视觉芯片类似，HERO 可以完成像素级并行处理。这种二维并行处理方式在完成 2D 图像滤波、背景减除、FAST、SIFT 特征点^[49]、LBP 算子^[30]等算法时体现出强大的性能。这些算法在很多的检测、识别应用中极为重要。如果它们在并行度较低的通用处理器上实现的话会消耗相对较长的时间。通过使用大量并行（Massively Parallel）的二维阵列，可以大大减少处理此类算法的时间，相关算法的 2 维并行实现方式将在第四章详述。

块并行处理时，需要执行包括方向判断（Orientation Assignment），局部特征提取，块匹配（Block Matching）以及直方图统计等多种复杂运算，因此需要更高的灵活性和复杂度。在 HERO 架构中，我们基于 MIPS 处理器架构设计了更加具有可编程性的 PPU 处理器。在像素级并行处理和块并行处理之间存在如图 2.14 中虚线所示的对应关系：块处理是在像素级并行

处理的结果之上进行的，所处理的图像为局部图像块，这一对应关系正如 HERO 架构中一个 PPU 对应 16×32 个 PE 的关系，这种对应关系较之前视觉芯片架构中的 PE-RP 对应关系是显著不同的。正如我们在架构设计考虑一节中分析得出，局部特征在计算机视觉中扮演着重要角色^[5, 39, 42, 50, 51]，局部特征主要是指图像局部某种统计特征，一般是在图像局部块中通过提取直方图等操作来完成。本文中的 PE-PPU 对应关系就是为了提高视觉芯片局部特征描述能力而提出来的。PPU 和 PE 阵列共享存储器，这也就意味着在 PE 阵列完成了像素级并行处理之后，PPU 就可以访问 PE 阵列中所对应的图像块而进行局部特征构建，这样的设计消除了以往视觉芯片中 PE-RP 设计存在的数据传输瓶颈。

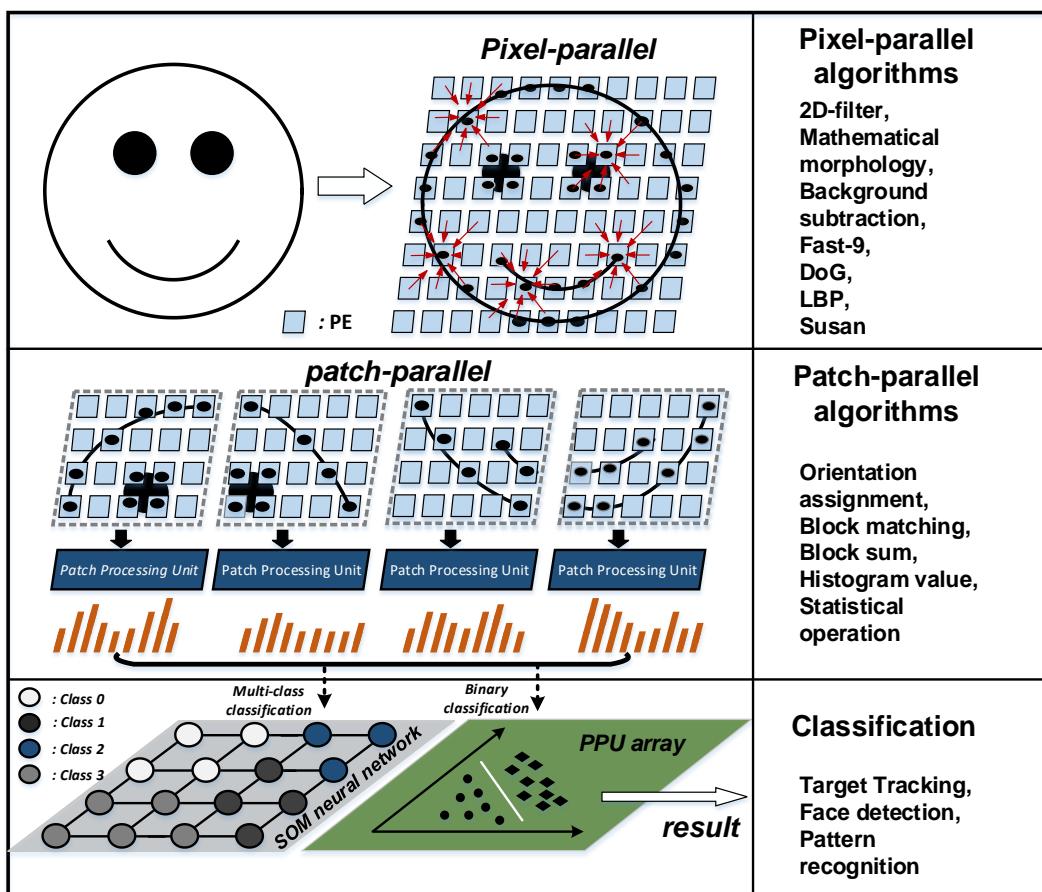


图 2.15 在 HERO 架构上完成视觉处理的一般流程以及相应算法

分布式并行计算主要用于快速实现 AdaBoost 分类和 SOM 神经网络分类。进行 AdaBoost 分类时，每个 Lane 计算一个或者多个弱分类器，所有弱分类器的分类结果经过并行计算可以迅速得到，这些结果经过简单的加权求和，就可以得到最终分类结果。Lane 同时也可以作为 SOM 神经网络的神经元使用，每个神经元可以存储 K-维的参考向量，并且可以通过学习向量量化（Learning Vector Quantization, LVQ）的方法进行在线训练和参

考向量更新。

图2.15所示,为使用 HERO 构架完成视觉处理的流程。首先,利用 PE 阵列完成如滤波、形态学、背景减除、特征点检测、特征生成等可像素级并行的算法,然后 PPU 访问存储于 PE 子阵列中的图像块并提供程序控制指令执行局部特征提取算法得到局部统计特征,所有 PPU 同时工作以并行处理所有图像块。最后,利用分布式并行计算实现 AdaBoost 或者 SOM 算法,对提取出的特征进行分类。MPU 可以根据分类结果和相关数据进一步做出决策以决定后续需要执行的操作。

2.3 新型视觉芯片架构特点

2.3.1 块并行处理

先前报道的视觉芯片架构是难以实现图像块并行处理的, HERO 架构中,每个 PPU 和 16×32 个 PE 形成紧密耦合,在这些 PE 完成图像块的处理后,PPU 可以立即通过共享存储器对图像块进行访问。所有 PPU 可工作于 SIMD 模式下,也就是通过同时对所有图像块进行处理来实现块并行。图 2.16 (a) 比较了串行与块并行两种处理的不同。在串行处理模式下,图像块按顺序被处理,算法必须遍历整个图像中所有的图像块,因此完成算法的时间与图像块的数目成正比。在块并行的处理模式下,所有的图像块被同时处理,算法只需要被执行一次就结束了对整个图像的处理,而该过程实现了于并行处理图像块数目相同的加速比。

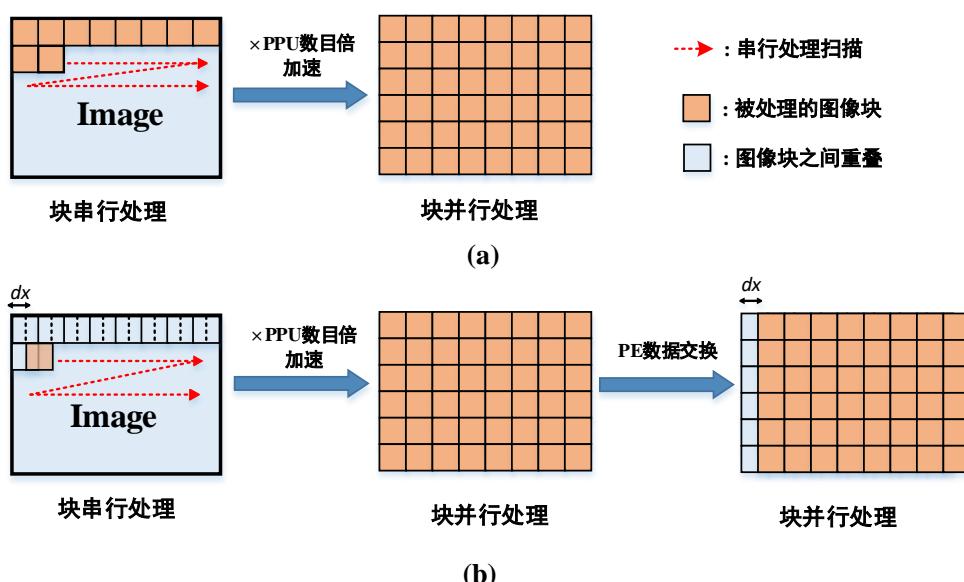


图 2.16 块串行处理与块并行处理的比较

进一步的，在检测和识别任务中，相邻图像块之间往往要求有一定的交叠以保证任何位置的特征都能被检测到，如图 2.16 (b) 中所示，俩相邻的块之间存在 dx 像素的交叠。在 HERO 架构中，借助 PE 之间的数据移动能力，可以高效的实现块交叠处理。只需要数个时钟周期，我们就能实现所有图像块中的数据移动 dx 像素。然后，我们在新的图像块中重新执行算法。

2.3.2 分布式并行分类

分类是图像处理与计算机视觉中的最为重要的问题之一，兼具高正确分类率和高性能的分类器实现对视觉系统至关重要。HERO 架构可以通过分布式并行的方式实现快速准确的分类。式 (2-2) 和 (2-3) 分别所示的 AdaBoost 和 SOM 分类器就可以在 HERO 架构中快速实现。这里，我们重现它们为式 (2-5) 和 (2-6)：

$$C(x) = \text{sign}(\sum_{t=1}^T a_t h_t(x) - T) \quad (2-5)$$

$$Dist_{i,j} = \text{argmin} \| FV - RV_{i,j} \| \quad (2-6)$$

AdaBoost 分类器是一种二分类器，它由 T 个弱分类器 $h_i(x)$ ($i=0,1,2\dots T-1$) 构成，弱分类器的输入为某一特征向量，而输出则为逻辑值 0 或者 1。SOM 神经网络计算输入特征向量和参考向量的距离，最终的分类结果与获胜神经元的位置 (i, j) 相对应。在 HERO 架构中，我们利用分布式并行的方式来实现上面两个分类函数。对 AdaBoost 而言，我们可以将式 (2-5) 重写为 (2-7) 和 (2-8)：

$$C(x) = \text{sign}(\sum_{j=0}^m partial_sum - T) \quad (2-7)$$

$$partial_sum_j = \sum_{i=j^k}^{j^k+k} h_i(x)^* a_i \quad (2-8)$$

其中 m 表示 HERO 架构中 Lane 的数目， k 表示每个 Lane 中计算的弱分类器的个数。很显然， m 个 $partial_sum$ 变量可以用 m 个 Lane 并行计算。最后 MPU 计算式 (2-8) 得到最终的分类结果。我们也可以将神经元参考向量 $RV_{i,j}$ 存储在 Lane 中，当要计算式 (2-6) 时，我们向所有 Lane 广播特征向量 FV ，各个 Lane 则并行计算参考向量与特征向量的距离。

硬件专用分类器往往消耗可观的面积，本文所提出的架构中，AdaBoost 和 SOM 神经网络分类器在没用使用任何额外硬件的情况下即可实现。AdaBoost 和 SOM 神经网络可以在不同的场景中使用。如果需要进行准确的正负类别分类，HERO 架构可以使用 AdaBoost 分类器，而 SOM 神经网络则可以用来进行多类别分类和在线更新。

2.3.3 灵活的图像块选取方式

为解决图像传感器成像质量差、像素和 PE 单元之间映射方式单一的问题，较为先进的视觉芯片都采用了像素-PE 分离阵列结构^[31, 48]。然而，该结构仍然存在以下不足：在这些视觉芯片中，一个 $M \times M$ 的 PE 阵列每次仅能采样图像传感器中 $M \times M$ 像素。但是，如果该 $M \times M$ 区域并非感兴趣区域，或者图像中存在多个感兴趣区域，那么处理器就需要从图像传感器后续读出的一帧或者多帧当中来获得感兴趣的区域。显而易见，所处理的图像大小受制于 PE 阵列大小是不合理的，这种方式限制了系统的灵活性并且严重减低了处理性能。在存储空间允许的情况下，HERO 架构中每个 MP 或者 VP 可以处理任意大小的图像块。

如图 2.17 (a) 所示，对于任意大小的像素阵列，可以通过 1:1 采样的方式得到图像传感器中的图像块 0 和图像块 1，这两局部图像块可以存储在两个 PPU 的局部存储器中并被并行处理。如图 2.17 (b) 和 (c) 所示，使用不同的采样间隔并选取不同的图像块，可以获得不同的图像块以满足视觉处理的要求。每个 PPU 的存储器中存储的图像块数目和图像块大小主

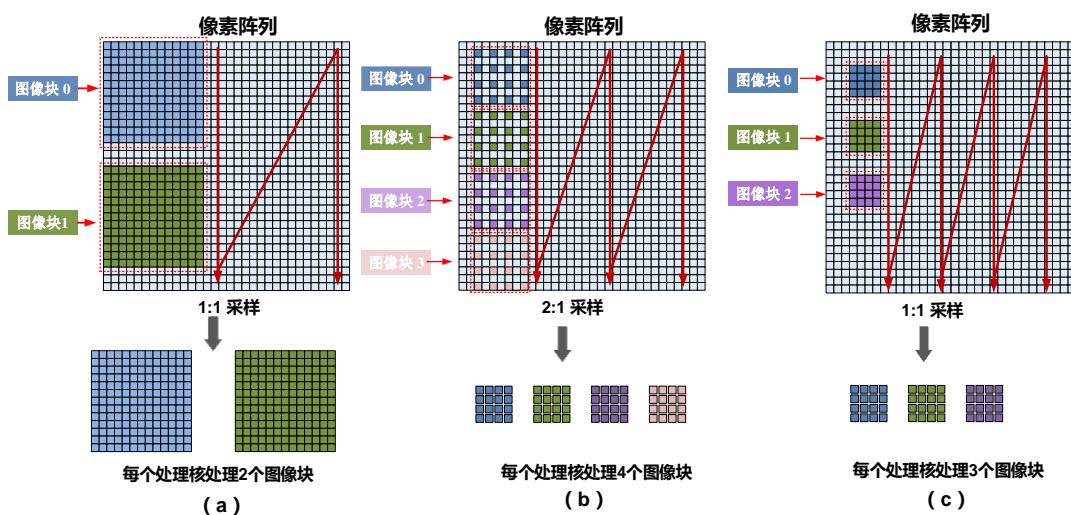


图 2.17 每个处理核可以处理任意大小的图像块

要收限于局部存储器的的大小。我们可以使用图 2.17 中红线所示 Z 字形扫描的方法，将图像中的图像块均匀的分配到 HERO 架构中的 8 个 PPU 的存储器中，因此只需要一次图像读出，整个像素阵列的数据都可以被处理，而不再局限于某一个区域或者某一大小。

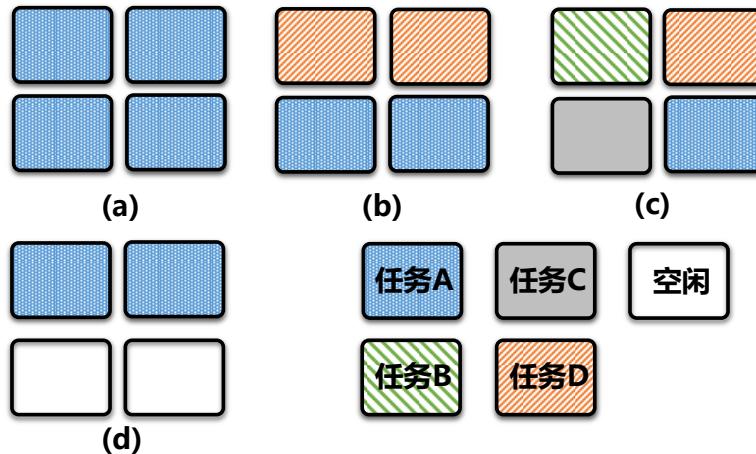


图 2.18 HERO 架构的多指令多数据模式

2.3.4 MIMD 工作模式

HERO 架构中，所有的计算簇具有独立的取指令能力，通过 MPU 对各个计算核心中 MP 的配置，使每个计算核心可以进行不同的工作。如图 2.18 所示，为 HERO 架构中计算核心的几种不同的工作模式。每个计算核心中的 MP 和与共享指令的 VP 完全相同的任务，但是其他计算核心可以进行完全不同的工作。比如，对于单一目标进行检测，可以使用图 2.18(a)的工作模式，所有计算核心用于完成追踪算法，对于多个目标的追踪，则可以使用图 2.18(b)或者(c)中的工作模式，不同的计算核心用于追踪不同的目标。如果所进行的视觉处理相对简单，在处理速度允许的情况下，可以使用图 2.18(d)的工作模式，只有部分计算核心工作，而另一些计算核心则处理不工作状态。

使用 MIMD 还可以实现传感器和处理器之间的流水工作。一般而言，图像传感器曝光和读出会消耗数毫秒的时间，即使对于 1000 帧每秒的高速图像传感器而言，读出过程也需要 1 ms，然而很多视觉芯片的应用算法运行时间小于 1ms，如果必须等待图像传感器曝光和读出完成之后才能开始处理，那么视觉芯片的系统性能将大大降低。借助计算核心可 MIMD 工作的特点，HERO 架构采用了如图 2.19 所示的传感器-处理器流水处理方式来掩盖图像传感器曝光和读出的时间。图像传感器每读出一段时间就会启

动一个计算核心对已经读出图像中包含的图像块进行处理，并持续该过程直到一帧读出结束。在下一帧来临之前，最先启动的计算簇一般已经完成了对上一帧图像中图像块的处理，因此可以流水的处理即将到来的图像块。

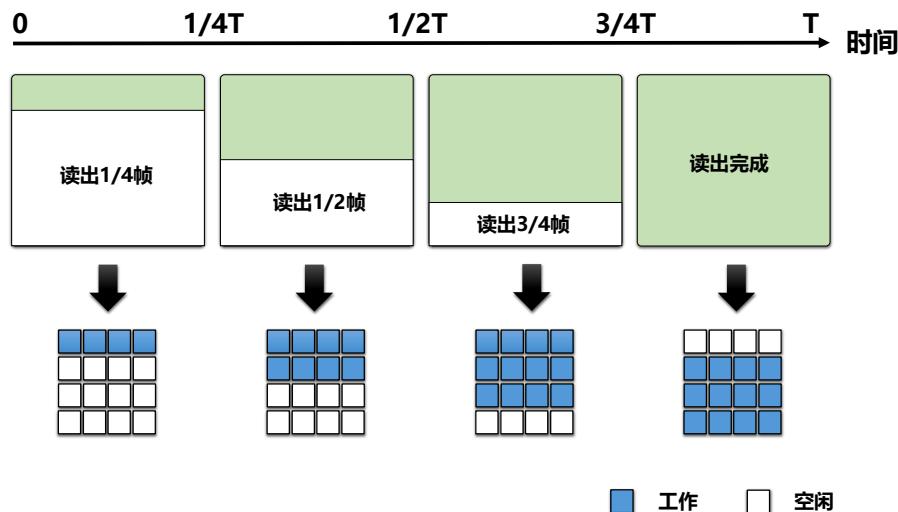


图 2.19 传感器-处理器流水工作

2.4 本章小结

本章分析用于视觉信息处理的传统方法和现代方法，并简要介绍了现代视觉处理框架中的三个关键步骤，然后从硬件实现的角度分析了其中包含的并行度和复杂度，最后提出适用于现代计算机视觉算法的基于多级异构并行处理的视觉芯片架构，该架构具备像素级、块级以及分布式并行处理能力，能够进行灵活的图像块选取并可工作于多指令多数据模式。

第3章 指令集以及关键电路设计

本章将实现 HERO 架构中的指令集和关键电路设计。首先，将介绍 HERO 架构的专用指令集以及指令集特点，随后，将详细介绍 PPU 处理器设计、Lane 设计、PE 单元电路设计以及缓存电路设计。

3.1 指令集设计

HERO 架构中 MPU 微处理器的指令集为 ARM 指令集，在这里我们不再介绍。本文主要设计了一套类 MIPS 指令集，并针对视觉处理做了相应扩展。我们首先简要介绍 MIPS 指令集(Instruction Set Architecture, ISA)。MIPS ISA 中包括三种指令类型，分别是 R-类型，I-类型以及 J-类型。R-, I- 和 J-type 分别代表了寄存器类型(Register)，立即数类型(Immediate)以及分支跳转类型(Jump)。寄存器类型的指令是指 MIPS 处理器进行运算的时候，两个操作数都来自于寄存器组中，R-类型的指令在三种指令类型

表 3.1 MIPS 指令集格式

R-type	op	rs	rt	rd	shamt	funct
	6 bits	5 bits	5 bits	6 bits	5 bits	6 bits

I-type	op	rs	rt	Constant or address
	6 bits	5 bits	5 bits	16 bits

J-type	op	Jump-target		
	6 bits	26 bits		

中最为复杂。如表 3.1 所示，R 类型的指令将 32 比特的指令分割为多个指令字段(Field)，其中每个指令字段所代表的含义如下：

- op：指令执行的基本操作，也被称为 opcode，指令操作码；
- rs：第一寄存器源操作数；
- rt：第二寄存器源操作数；
- rd：目的寄存器；
- shamt：指令进行逻辑算术移位时的位移量；
- funct：指令执行进行的运算，功能码；

立即数类型的指令是指 MIPS 处理器处理的两个操作数中一个来自于寄存器组，而另一个则为立即数。从表 3.1 可以看出，立即数的最大长达为 16 位。J 类型的指令在 MIPS 处理器需要跳转到新的程序地址开始执行操作时使用，其中跳转的最大偏移量为 26 位。

HERO 指令集在 MIPS ISA 的基础上进行了部分修改，如表 3.2 所示，指令类型分为四类，分别为寄存器类型、立即数类型、L-1 类型以及 L-2 类型。其中寄存器类型指令和立即数类型指令主要用于对 PPU 进行编程，L-1 和 L-2 型指令用于对 PE 和 Lane 进行编程，并可以完成 PPU 和 PE 与 Lane 之间的数据交互。寄存器类型指令和立即数类型指令基本和原有 MIPS 指令集类似，但是缩短了 rs, rt 以及 rd 的比特位数，并且增加了 format 字段。在 HERO 架构中，PPU 处理器的寄存器组中只有 16 个寄存器，因此 rs, rt, rd 以及 shamt 字段使用 4 位比特数就已足够，format 字段为两比特，用于区分这四类指令，最后 2 比特目前没有使用。立即数类型

表 3.2 HERO 指令集

HERO R-类型指令							
format	op	rs	rt	rd	shamt	funct	null
2 bits	6 bits	4 bits	4 bits	4 bits	4 bits	6 bit	2 bits
[31:31]	[29:24]	[23:20]	[19:16]	[15:12]	[11:8]	[7:2]	[1:0]

HERO I-类型指令				
format	op	rs	rt	Constant or address
2 bits	6 bits	4 bits	4 bits	16 bits
[31:31]	[29:24]	[23:20]	[19:16]	[15:0]

HERO L-1 类型指令								
format	op	rs	rt	rd	cond	dire1	funct	dire2
2 bits	6 bits	4 bits	4 bits	4 bits	3 bits	1 bit	6 bits	2 bits
[31:31]	[29:24]	[23:20]	[19:16]	[15:12]	[11:9]	[8]	[7:2]	[1:0]

HERO L-2 类型指令								
format	op	rs	rt	rd	cond	shamt1	funct	shamt2
2 bits	6 bits	4 bits	4 bits	4 bits	3 bits	1 bit	6 bits	2 bits
[31:31]	[29:24]	[23:20]	[19:16]	[15:12]	[11:9]	[8]	[7:2]	[1:0]

指令与此类似，其中立即数长度为 16 比特。HERO 作为 SoC 系统，其程序存储器的空间不会非常大，所以指令集中没有设计长跳转指令，所有的程序跳转都通过可以使用分支跳转语句来实现。L-1 和 L-2 类型指令中几个指令段的意义如下：

- cond：条件操作码，表示该条指令所得到的结果是否写回寄存器由条件寄存器的状态决定。表 3.3 中所示为由 cond 字段控制的八种不同条件操作；
- dire1, dire2: dire1 与 dire2 组成的 3 比特码用于指定某个邻近 PE，并将其该 PE 的 rt 寄存器数据作为本 PE 的操作数；
- sham1, sham2: sham1 与 sham2 组成的 3 比特码用于指定寄存器操作数的移位偏移量。

表 3.3 Lane 电路支持的八种条件操作

条件操作	执行写操作	条件寄存器状态
AL	任何情况下执行	任意状态
NA	任何情况下不执行	任意状态
EQ	相等	$Z == 1'b0$
NE	不相等	$N != 1'b0$
CS	大于等于或者非符号数 ≥ 1)	$C == 1'b1$
CC	小于或者非符号数 $<$	$C == 1'b0$
MI	负数	$N == 1'b1$
PL	正数或者零	$N == 1'b0$

L-1 类型指令主要用于 Lane 与 Lane 或者 PE 与 PE 之间的数据交换以及 PE 阵列的控制。它在诸如图像滤波、形态学操作、纹理特征生成等操作时显得非常有用。利用条件运算操作，Lane 可以完成 C 语言中 *if else* 语句。执行图 3.1 中所示的程序时，可以首先比较 a, b 的大小，然后在下一个周期计算 $c+1$ 的结果，并使用表 3.1 中的 CS 作为写回条件将计算结果写

```

if (a >= b)
    c = c + 1;
else
    c = c;

```

图 3.1 一段包含 *if else* 的程序

回寄存器。如果 $a >= b$ 的条件满足，那么 $c+1$ 的就被写回寄存器，反之，寄存器中 c 变量的值保存不变。条件操作的电路包含 3 个标志寄存器，分别为是否为零标志位 Zero, 是否为负标志位 Negative 以及进位标志位 Carry。

表 3.4 一些具有代表性的指令

<i>name</i>	<i>action</i>	<i>name</i>	<i>action</i>
<i>add</i>	$pr3 = pr1 + pr2$	<i>sub</i>	$pr3 = pr1 - pr2$
<i>sll</i>	$pr3 = pr1 >> imm$	<i>mv</i>	$pr2 = pr1$
<i>and.vv</i>	$pr3 = pr1 \& pr2$	<i>or.vv</i>	$Pr3 = pr1 / pr2$
<i>mvlef.3</i>	$pr3 = pr1$ of the 3 rd left PE	<i>mvrg.1</i>	$pr3 = pr1$ of the 1 st right PE
<i>ld</i>	$pr3 = MEM[cr]$	<i>st</i>	$MEM[cr] = pr3$
<i>mv1</i>	$pr3 = imm$	<i>bc</i>	$pr3 = cr$
<i>sltu</i>	$pr3 = 1$ if $pr1 < pr2$	<i>rdu</i>	<i>Is in pr1</i>

L-2 类型指令用于 Lane 内数据计算。表 3.4 列出了一些具有代表性的指令，其中 pr^* 以及 cr^* 分别表示 Lane 和 PPU 中的通用寄存器。表中，*add*, *sll*, *sub* 等无后缀的算术运算指令与 MIPS 指令集中寄存器类型指令意义相同。比如，*add* 表示两个寄存器操作数相加并将结果存储到寄存器中，*sll* 则表示逻辑左移。具有 *vv* 后缀的指令则表示该指令进行按位运算，用于进行 PE 阵列操作，这类指令可以完成逻辑与、或、非以及异或等操作。例如，*and.vv* 指令就是将寄存器中操作数 *pr1* 和 *pr2* 按位进行逻辑运算，*or.vv* 则按位进行或运算。*mvlef* 以及 *mvrg* 指令表示将临域 Lane 中某个寄存器值复制到当前 Lane 寄存器中，指令后缀的数字即上述 *dire* 指令字段，用于指明具体是哪一个临近 Lane。Lane 的数据载入和存储分别使用 *ld* 和 *st* 两条指令，*ld* 指令将 PPU 中寄存器 *cr* 的值作为存储器地址，并将该地址以及其后连续的 32 个数据载入至 32 个 Lane 的寄存器中。*st* 指令将 32 个 Lane 的寄存器 *pr* 中的值存储到以 PPU 中 *cr* 寄存器值为起始地址的连续存储空间中。*mv1* 指令对 Lane 寄存器 *pr* 进行立即数赋值，*bc* 则将 PPU 寄存器中的值广播到所有 Lane 的寄存器中。*rdu* 指令返回当前 Lane 寄存器操作数中 1 的个数。

指令集实现了像素级并行、块级并行以及分布式并行处理所需的操作，具有指令跳转、寄存器寻址、多比特运算等一系列经典视觉芯片指令集中所不具备的指令，使得整个系统的可编程性与易编程性大大提高。表 3.2 中对 PE、Lane 以及 PPU 的控制以及运算指令均首先被 PPU 解码，同时考虑到 PE 和 Lane 事实上为同一硬件电路，所以下面几个小节，我们将按照 PPU 电路、Lane 电路以及 PE 电路的顺序对该指令集的电路实现进行介绍。

3.2 PPU 电路设计

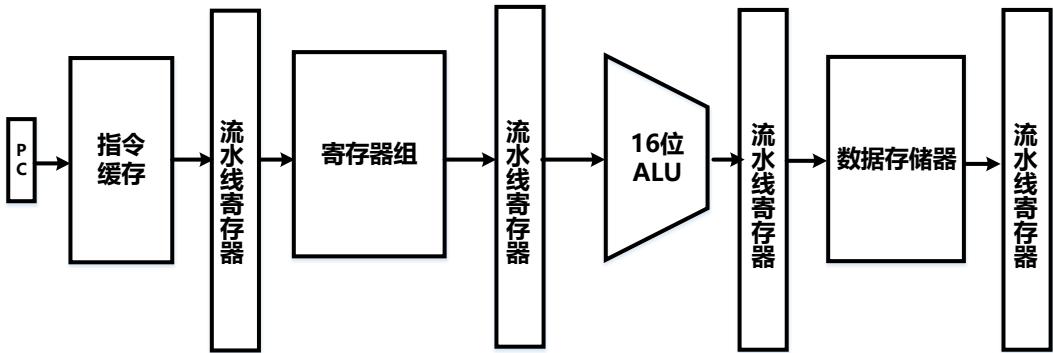


图 3.2 五级流水 PPU 处理器架构

HERO 架构中的块处理器采用了经典 MIPS (Microprocessor without Interlocked Pipeline Stages) 处理器架构，MIPS 源于 Stanford 大学 80 年代初 VLSI 研究计划的一部分，其设计者 John Hennessy 教授认为通过指令管线化 (Instruction Pipelines) 可以提高处理器运算的性能，并与 1985 年设计出第一代 MIPS 处理器。因为电路结构简洁且性能卓越，MIPS 处理器被广泛运用于嵌入式系统、游戏机、服务器等多个领域，目前智能终端中所使用的 ARM 系列处理器也是由 MIPS 处理器发展而来，MIPS 处理器是最早，最成功的处理器之一。

图 3.2 为本文 PPU 块处理器所采用的 MIPS 电路架构图，它由指令缓存、寄存器组、16 比特 ALU，数据存储器和多个流水线寄存器组成。指令缓存用于存储指令，这里的数据存储器即是与 Lane 共享的局部存储器。控制 PPU 操作的每个指令都必须经过如下五级流水：

1. 取指令(Instruction Fetch, IF):

使用程序计数器 (Program Counter, PC) 的值作为地址，从指令缓存中取出指令；

2. 指令译码和读取寄存器(Instruction Decode, ID):

根据上一步中读取的指令分析该指令的类型和及其操作类型，并给出相应的控制信号。同时根据指令中的给出的寄存器地址访问寄存器堆并获得相应的寄存器值；

3. 执行指令(Execution, EXE):

根据指令译码阶段所译码的控制指令或者操作类型进行计算，例如，对算术类型的指令完成算术运算，对存取类指令，则计算地址；

4. 存储器访问(Memory, MEM):

访问数据存储器，只对访问存储器类型的指令（主要是 Load/Store 指令）起作用，其他类型的指令在这个流水段不会做任何事情；

5. 写回(Write Back, WB):

如果指令存在目的寄存器，则将指令最终的结果计算结果写回到目的寄存器中。

指令译码阶段将产生控制该指令后续操作的所有控制信号，这些控制信号在 PPU 中的命名和作用如表 3.5 所示。这 12 个控制信号将随着流水线寄存器一直传递，直到产生这些控制信号的指令执行结束，其中前 7 个控制信号控制 PPU，而后面 5 个控制信号被传递至 Lane。

表 3.5 PPU 解码产生的控制信号

控制信号	控制信号为 0 时	控制信号为 1 时
RegDst	目的寄存器为 rt 指令段指定	目的寄存器由 rd 指令段指定
RegWrite	无任何作用	计算结果将写回目的寄存器
ALUSrc	ALU 的第二操作数由 rt 指定	ALU 的第二操作数为立即数
PCSrc	下一条指令的地址为 PC+4	下一条指令的地址由 ALU 计算得到
MemRead	无任何作用	读数据存储器
MemWrite	无任何作用	写数据存储器
MemToReg	写入寄存器的值由 ALU 计算得到	写入寄存器的值来自存储器
Lane_BroadcastToReg	无任何作用	PE 寄存器的写入值由 PPU 广播得到
Lane_ALUSRc	PE 第二操作数由 rt 指令段指定	PE 第二操作数来自邻近 PE 的寄存器
Lane_MemToReg	写入 PE 寄存器的值由 ALU 计算得到	写入寄存器的值来自存储器
Lane_MemWrite	无任何作用	写数据存储器
Lane_MemRead	无任何作用	读数据存储器

由于采用了管线化的设计，在每个时钟周期，MIPS 处理器中都同时存在 5 条不同的指令在执行。如果指令间存在一定的关联性，则在执行过程中可能会出现数据、结构以及分支冒险。如图 3.3 所示，如果第二、三、四条指令在解码过程中需要用到的寄存器值由第一条指令写回，那么第一条指令和后续 3 条指令间就存在数据冲突，后面三条指令解码开始时，第一条指令还没有开始写回。数据冲突会降低 PPU 处理器的性能，解决这一问题的方法是采用前递（Forwarding）技术。前递技术的具体实现如图 3.4 所

示，在流水线 ALU 前通过多路选择直接把前面指令的运算结果作为后面指令的输入。多路选择器由前递逻辑电路控制，该电路判断指令间的相关性，并产生相应的信号来选择将 MEM 或者 WB 两级流水线中计算结果送到 ALU 中。本文还设计了冒险检测以及 Stalling 等方法来有效减小这几种冒险对 PPU 处理器性能的影响。

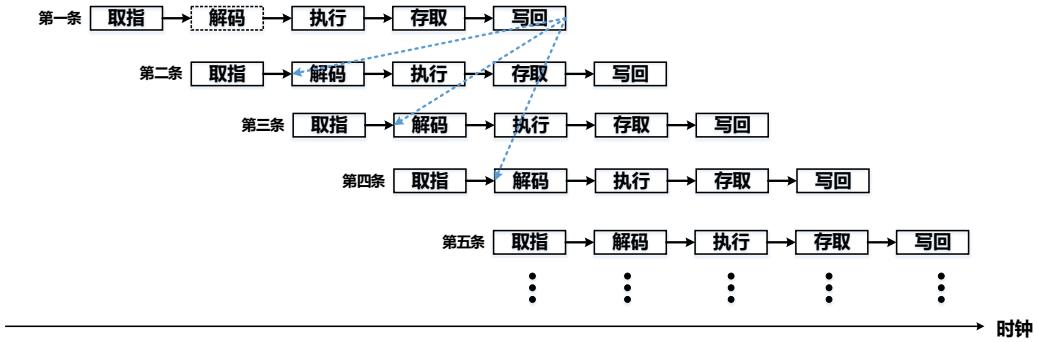


图 3.3 流水线冲突

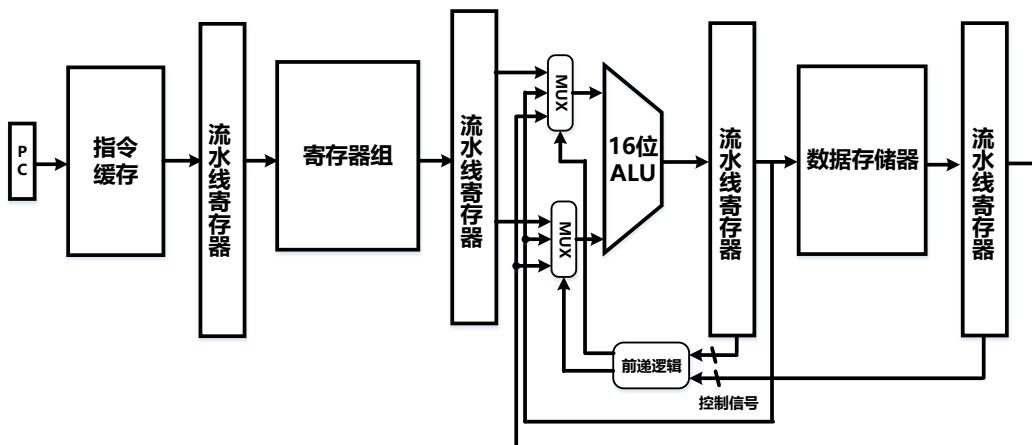


图 3.4 使用前递计算的处理器架构

为保证一个时钟周期内能够获得存储器中的数据，HERO 架构中指令缓存、数据存储器都和处理器所使用的时钟存在 180 相位差，且所有部件均在时钟上升沿采样。如图 3.5 所示，PPU_clock 和 memory_clock 分别为 PPU 处理器和存储器使用的时钟。当进行存储器读操作时，程序计数器（Program Counter, PC）和流水线寄存器在 PPU_clock 时钟上升沿结束后向指令缓存和数据存储器给出正确的读地址，相隔半个周期之后，指令缓存和数据存储器在 memory_clock 的时钟上升沿采样到正确的读地址，经过小于半个时钟周期的短暂延迟，数据端口 rd_data 出现正确的数据，在

下一个 PPU_clock 的时钟上升沿，该数据被后一级寄存器读走。当进行写操作时，在 PPU_clock 的时钟上升沿后，写地址和写数据稳定，再经过半个时钟周期，该写地址以及数据值被 memory_clock 的上升沿读到。

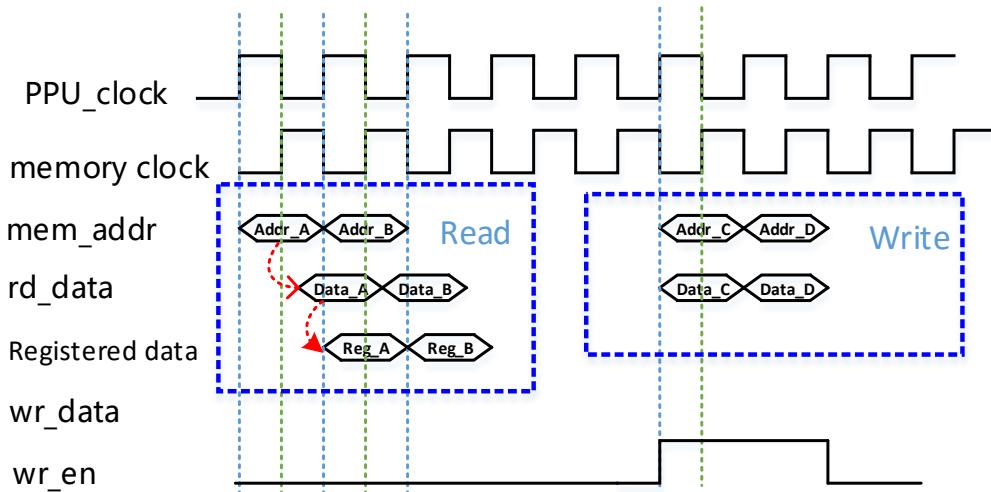


图 3.5 处理器与存储器时钟 180 度相位差时序

PPU 电路设计具有很高的灵活性，通过编程可以完成一般视觉处理所需的控制、算术运算操作。采用如图 3.6 所示的程序，PPU 可实现在 SIFT、SURF 以及 HOG 算法中的像素梯度方向确定。在图 3.7(a)中，根据像素在 x 和 y 方向上的梯度(dx, dy)，梯度方向可划分至 $0 \sim 7$ 的八个梯度区间之一。图 3.7(b)中，如果 $dx > 0$ 成立，那么可以判断梯度方向应该落在 $0 \sim 3$ 区间之内，进一步的，如 $dy > 0$ 也成立，那么该点的梯度方向应该属于 0 或者 1 之间。最后，我们通过判断 $|dx| > |dy|$ 是否成立来判断该像素具体落在哪个梯度区间。

```

if (dx > 0 && dy > 0 && |dx| > |dy| )
    gradient = 0;
else if (dx > 0 && dy > 0 && |dx| < |dy| )
    gradient = 1;

```

⋮ ⋮ ⋮

图 3.6 利用 PPU 来计算梯度方向的程序

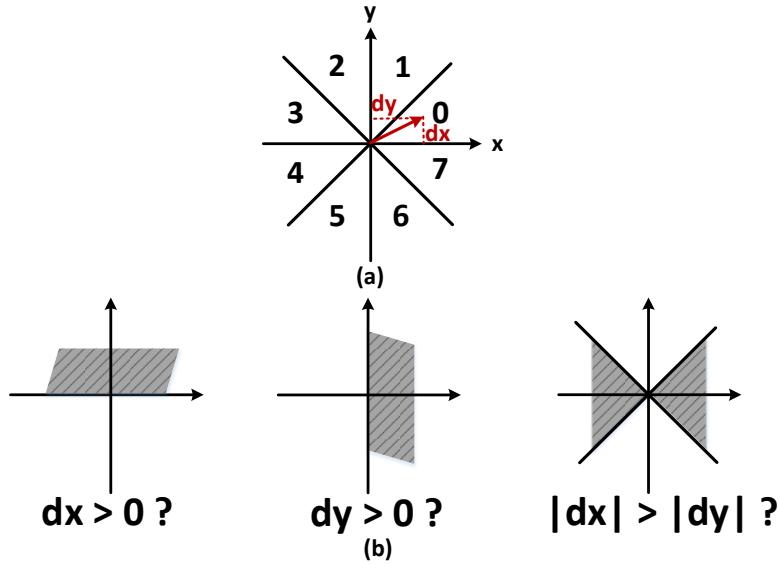


图 3.7 利用 PPU 完成梯度方向计算

3.3 Lane 电路设计

如图 3.8 所示为 HERO 架构中所采用的 Lane 设计电路图。它主要包含一个通用寄存器组，多个条件寄存器，16 比特 ALU 以及多个多路选择器，Lane 不使用独立的存储器，而是和一个 PPU 共享存储器。Lane 中如 MemToReg，Broadcast，Direct，ALUSrc 等控制信号都是通过 PPU 中指令解码器解码广播得到。ALU 在每个时钟周期接收两个操作数，其中一个来自于通用寄存器组，另一个则由通用寄存器组或者其临近 6 个 Lane 的通

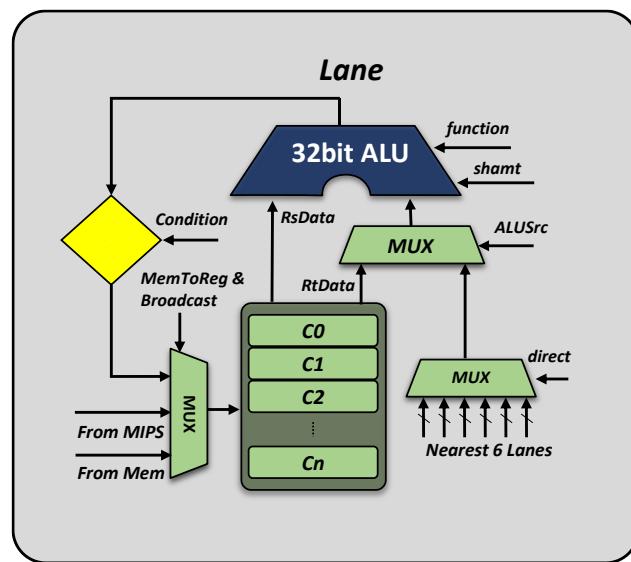


图 3.8 Lane 电路图

用寄存器组中选择。该方法可以快速实现临近 Lane 之间的数据交互。Lane 具备表 3.3 中的八种条件操作，可以执行简单 *if else* 语句的能力。Lane 的 16 比特 ALU 具有完善的逻辑功能，可以完成逻辑左移，逻辑右移，加减以及一些特殊运算，可以在每个时钟周期该完成一次计算。寄存器组每个周期根据指令中 Rs 以及 Rt 的值，可以相应的读出两个数据 RsData 和 RtData，同时每个周期可以从 PPU 广播值、存储器以及 ALU 输出中选择一个值写入寄存器。Lane 通过表 3.6 中的 Dire 字段编码来确定选择邻近 6 个 Lane 中的哪一个。

表 3.6 利用 Dire 编码实现邻近 PE 的选取

Dire 编码	选取的 Lane
001	左侧的第一个 Lane
010	左侧的第二个 Lane
011	左侧的第三个 Lane
100	右侧的第三个 Lane
101	右侧的第二个 Lane
110	右侧的第一个 Lane

每个 Lane 的操作都必须经过取指令、指令解码、以及执行三级流水来，其中取指、以及解码两级流水线在 PPU 中完成。最后一级流水—执行一大部分为寄存器到寄存器（register to register）操作，在一个周期内完成。对 Lane 在流水线上的优化，有利于系统主频的进一步的提高。

3.3.1 分布式并行处理

PE 可以完成分布式的并行计算。我们重写在 2.1.4 小节中计算分类函数子项的程序如图 3.9 所示，其中 a , b 和 c 为三个包含 T 个元素的数组。 $funct$ 一般为内积或者多项式运算， p 为保存子项计算结果的数组。如图 3.10 所示，在进行分布式并行的计算之前，将数组 a , b 和 c 存储于局部存储器中，其中， $a[0]$ 、 $a[1]$ 、 $a[2] \dots a[T]$, $b[0]$ 、 $b[1]$ 、 $b[2] \dots b[T]$ 以及数组 c 的各个元素按照位置相邻存储。进

```
for (int i = 0; i < T; ++i)
    p[i]=funct(a[i],b[i],c[i]....);
```

图 3.9 简单分类函数 C 语言代码

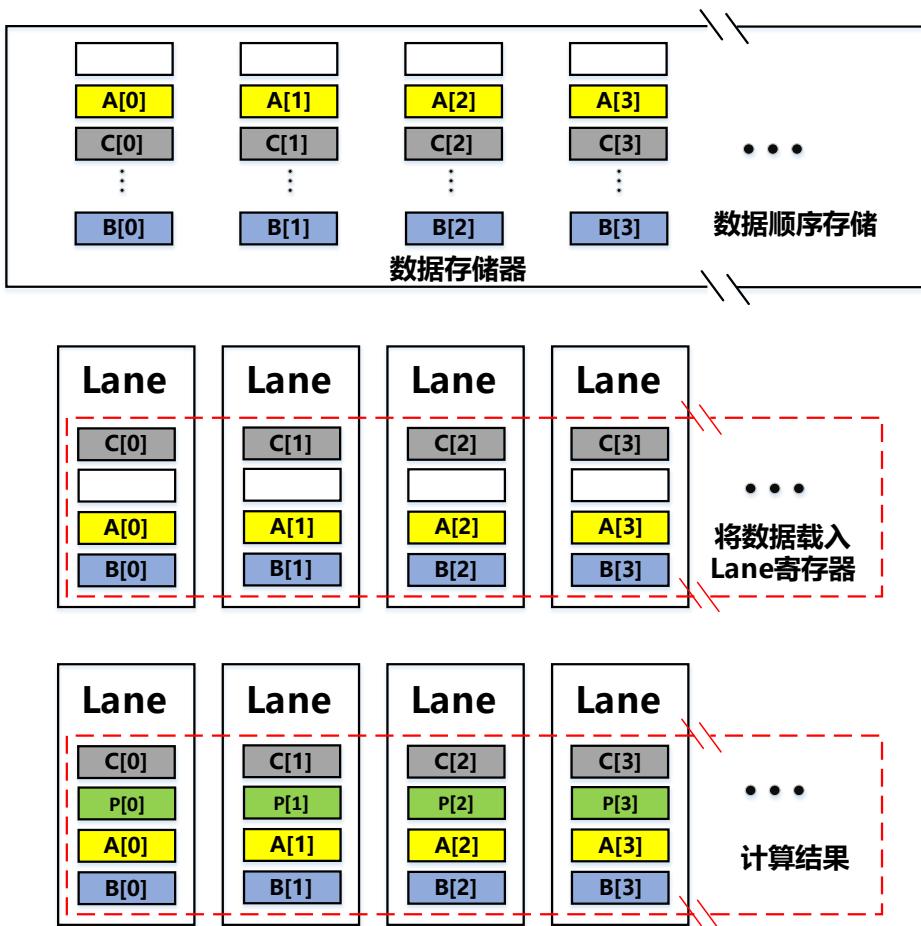


图 3.10 使用 Lane 完成分布式并行计算

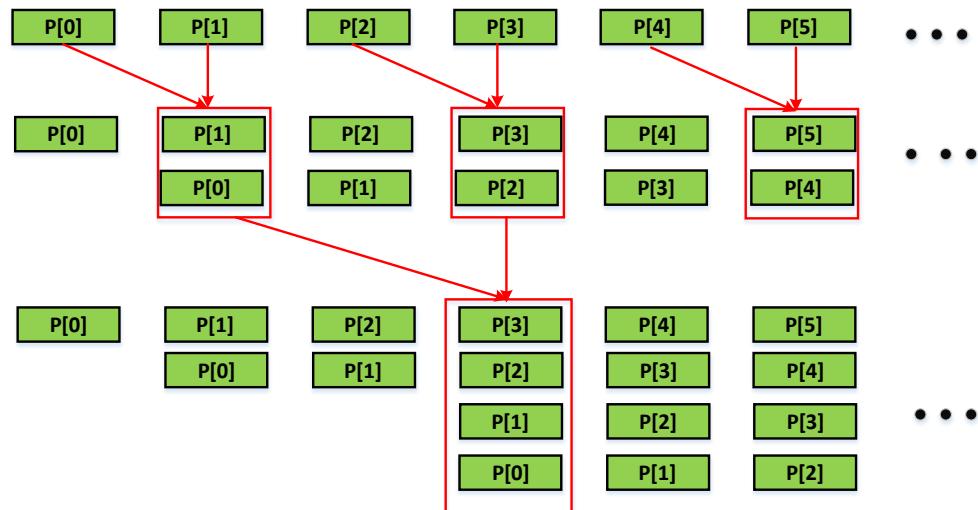


图 3.11 快速进行 Lane 间数据求和运算

行分布式并行计算时，首先使用 *ld* 指令载入 *a* 数组元素至 Lane 寄存器中，这样每个 Lane 寄存器寄存 *a* 数组中的一个元素。重复使用 *ld* 指令，直至数组 *b* 和数组 *c* 也分别载入至 Lane 寄存器中。此时，对于任意一个 Lane，其寄存器组中分别存储了 *a*, *b* 和 *c* 数组中的一个元素。使用 Lane 算术运算指令计算 $\text{funct}(a, b, c)$ 即可得到 *p* 数组中各个元素。得到 *p* 数组后，利用 Lane 间数据交互指令，可以快速数组元素之和。如图 3.11 所示，将相邻 Lane 中 *p* 数组元素相加，可以分别得到 *p* [0] + *p* [1] 之和，*p* [2] + *p* [3] 之和等等。继续使用 Lane 数据交互指令可以得到 *p* [0] + *p* [1] + *p* [2] + *p* [3] 之和以及 *p* [4] + *p* [5] + *p* [6] + *p* [7] 之和，以此类推便可快速的实现 *p* 数组中各个元素的求和。

3.3.2 快速直方图统计

图 3.12 所示为一种使用 Lane 完成直方图统计的方法，其中不同灰度圆形表示不同的数值。首先使用 PPU 处理器对 32 个 Lane 中 R_i 寄存器赋值。例如，对第一个 Lane 寄存器 R_i 赋值为 0，第二个 Lane 寄存器 R_i 赋值为 1，以此类推直至所有 Lane 中的 R_i 寄存器都被赋予不同的值。每一个所赋值代表进行直方图统计的某一通道（Bin 值）。 R_i 被赋值为 1 的 Lane 用于统计数据中 1 出现的次数， R_i 被赋值为 2 的 Lane 则统计数据中 2 出现的次数，以此类推。PPU 从存储器中读取需要进行直方图统计的数据，并通过广播指令将该值广播至 PPU 下 32 个 Lane 中，Lane 并行比较该值是否与其 R_i 值相同。如果相同，则该 Lane 的 R_j 寄存器加 1，反之 R_j 寄存器不变。

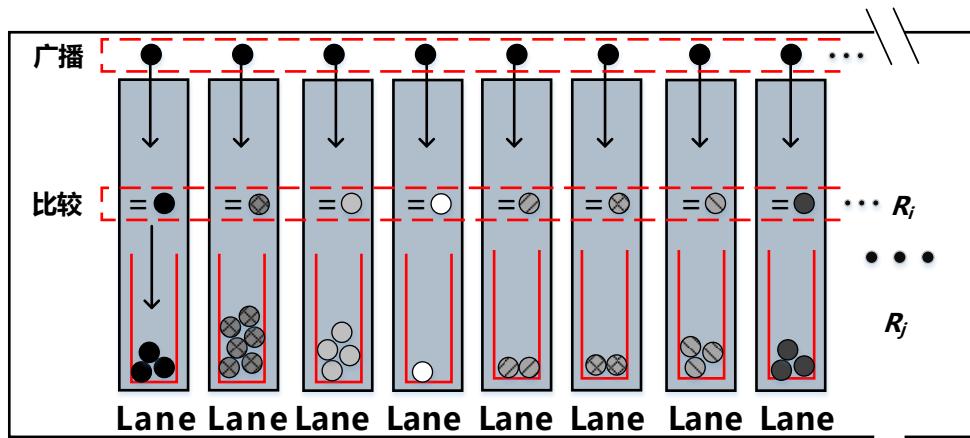


图 3.12 一种使用 Lane 加速直方图统计

图 3.13 为另一种利用 Lane 完成直方图统计的方法。在 PPU 对各个 Lane 的 R_i 赋不同 Bin 值之后，我们直接使用数据载入 *ld* 指令为每个 Lane 载入

待统计的数据。然后每个 Lane 并行比较该值是否与其 R_i 值相同，如果相同则 R_j 寄存器加一，反之不变。在图 3.13 (a) 中，每个 Lane 所载入数据与其 R_i 值都不相同，所以统计数据不发生变化。随后，数据循环左移，继续与 R_i 值比较并根据结果更新 R_j 寄存器的值。重复该过程直至每个 Lane 遍历所有数据，如图 3.13 (b) 和图 3.13 (c) 所示。

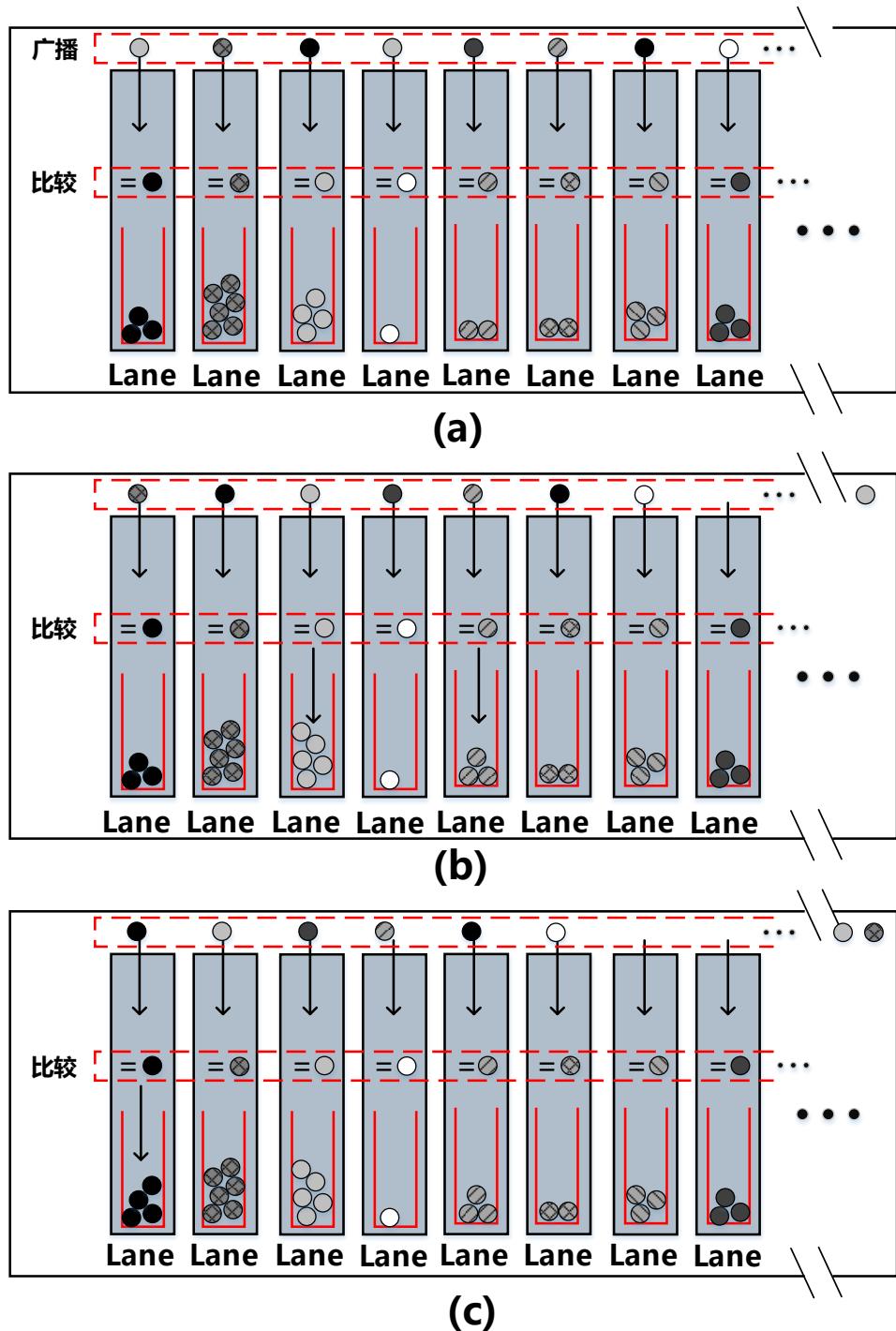


图 3.13 另一种利用 Lane 进行直方图统计的方法

时钟周期

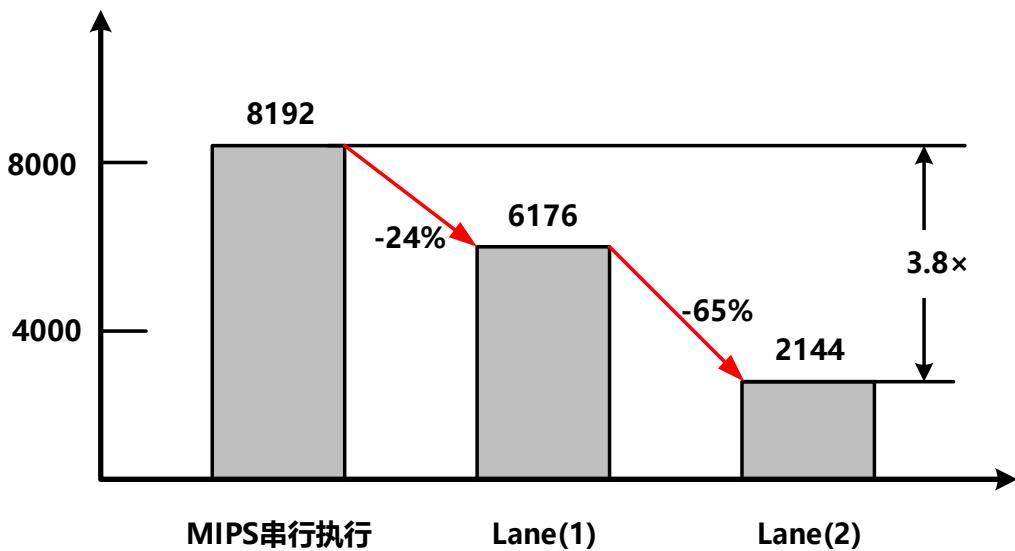


图 3.14 使用 PE 并行处理统计提高直方图统计效率

上述方式大大提高了直方图统计的执行效率。图 3.14 中对比了使用 MIPS 串行执行、图 3.12 以及图 3.13 三种方法进行 2048 个数据的直方图（32 个 Bin）统计所消耗的时钟周期数。使用 MIPS 进行串行处理时，首先需要将待进行统计数据取到寄存器 R_i 中，然后使用该值作为存储器地址并将该地址所对应的数值 $\text{Mem}[R_i]$ 载入 R_j 寄存器，对 R_j 加 1 并将结果存回 $\text{Mem}[R_i]$ 。重复上述过程，直至遍历所有数据即完成了直方图统计。该过程需要不断的进行存储器读写，统计过程没有并行，在不考虑程序控制等操作所消耗的时钟周期情况下，该方法耗时 8192 个时钟周期。使用图 3.12 所示的方法进行处理时，每个时钟周期从存储器中取得数据，经过 PE 并行计算后，结果直接存储在 Lane 寄存器中。虽然该统计方法使用了 Lane 并行计算，并且数据不需要立即存回存储器，但是在取数据的时候仍然是串行进行的。该方法相对于使用 PPU 串行操作节约了 24% 左右的时钟周期。图 3.13 所采用的统计方法，不仅利用了 Lane 并行计算和不需要立即将计算结果回存到存储器特点，并且在每个时钟周期，并行取得多个待处理数据，进一步提高了处理效率。该方法在第二种方法的基础上减少了 65% 的指令周期数量。图 3.14 中可以看出，使用第三种方法可以提高直方图统计的效率达 3.8 倍。

3.4 PE 电路设计

本文没有单独进行 PE 电路设计，而是直接采用 Lane 的电路来完成 PE 阵列所需的操作。如图 3.15 所示，为 PE 和 Lane 之间的对应关系，每个具有 16 比特运算能力的 Lane 可以等效的看做 16 个 PE。如图 3.16 所示为 Lane 电路等效的 16 个 PE 中第 i 个 PE 的电路逻辑，其中 1 比特 ALU 为 Lane 电路中 16 比特 ALU 的第 i 比特，它可以完成 1 比特加法、逻辑非、逻辑与、逻辑或操作。

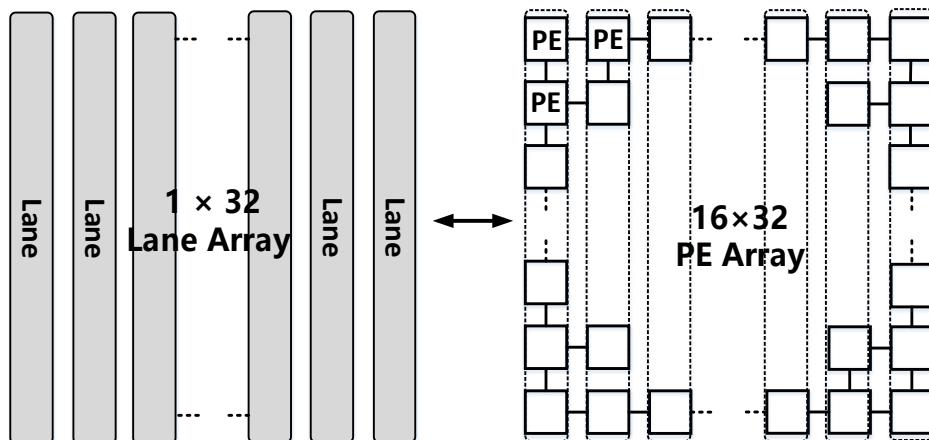


图 3.15 Lane 阵列和 PE 阵列之间的关系

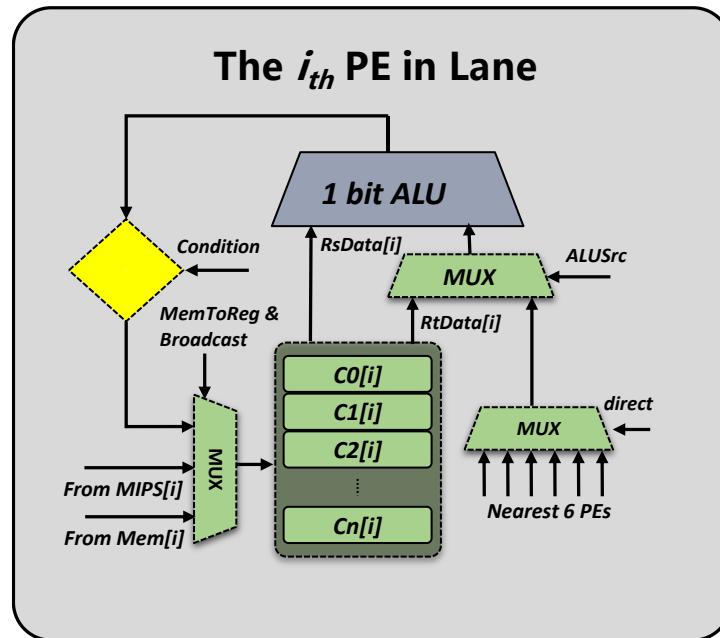


图 3.16 PE 等效电路图

作。图中虚线所示部分电路与图 3.8 中 Lane 电路相同，为所有 PE 共用。每个时钟周期，PE 的 ALU 读取两个 1 比特操作数，其中一个操作数为寄存器组读出数据 RsData 的第 i 比特 $RsData[i]$ ，另一个数据从 Lane 寄存器

读出的 RtData 中第 i 比特以及近邻 6 个 PE 中寄存器读出数据的第 i 比特中选择。

表 3.7 部分 PE 可完成的复杂运算

$c = a + b$	$c = a - b$
$c = \max(a, b)$	$c = \min(a, b)$
$\partial f / \partial x$ 或 $\partial f / \partial y$	$\partial^2 f / \partial^2 x$ 或 $\partial^2 f / \partial^2 y$
$c = a < b ? a : 0$	$c = a > b ? a : 0$
$c = a $	非极大值抑制

当 PE 进行像素级并行处理时，虽然完成 1 比特逻辑运算，但是通过如果级联多个按位逻辑操作，则可以完成多种复杂的多比特运算。表 3.7 列出了只使用按位运算指令时，PE 可以完成的复杂运算，其中 a, b, c 皆为多比特变量， $\partial f / \partial x$ 和 $\partial^2 f / \partial^2 x$ 分别是图像中的水平方向的一阶梯度和二阶梯度。从理论上讲，仅仅使用按位运算指令可实现的运算远远超过表中所列。下面以实现 $\max(a, b)$ 操作为例来说明如何使用按位运算指令来完成多比特复杂运算。首先，可以将 $\max(a, b)$ 转化为如下表达：

$$\max(a, b) = a > b ? a : b \quad (3-1)$$

即对变量 a 和 b 施加条件运算符来取得其中的较大的值。为使用按位运算指令实现这一功能，我们将上述式子进一步等效为式 (3-2)：

$$a > b ? a : b \rightarrow \text{sign_bit}(b-a) \& a + \text{sign_bit}(a-b) \& b \quad (3-2)$$

对于有符号数，两个数的大小可以通过减法之后所得结果的符号来判断。使用 $\text{sign_bit}(b-a)$ 来表示 $(b-a)$ 的符号位，如果 b 大于 a 则 $\text{sign_bit}(b-a)$ 等于 0，那么 $\text{sign_bit}(b-a) \& a$ (按位相与) 的结果为 0，而 $\text{sign_bit}(b-a) \& b$ 的结果为 b ，表达式 (3-2) 的最终结果即为 b 。如果 b 小于 a 则 $\text{sign_bit}(b-a)$ 等于 1，那么 $\text{sign_bit}(b-a) \& a$ (按位相与) 的结果为 a ，而 $\text{sign_bit}(b-a) \& b$ 的结果为 0，表达式 (3-2) 的最终结果为 a 。因此，通过式 (3-2) 我们就可以得到 a 和 b 中的较大值。为了使用按位运算指令实现 $b-a$ ，我们需要计算 $b+(-a)$ ， $-a$ 可以通过对各位进行取反，最后再加 1 的方式来得到， $\text{sign_bit}(a-b)$ 为 $\text{sign_bit}(b+(-a))$ 取反。式 (3-3) 为使用按位运算指令实现 $\max(a, b)$ 的最终方法：

$$a > b ? a : b \rightarrow \text{sign_bit}(b + -(a)) \& a + (\sim \text{sign_bit}(b + (-a))) \& b \quad (3-3)$$

类似的，我们可以得到 $\min(a, b)$ 在 PE 中的求解方法如式（3-4）：

$$a > b ? b : a \rightarrow \text{sign_bit}(b + -(a)) \& b + (\sim \text{sign_bit}(b + (-a))) \& a \quad (3-4)$$

使用上述逻辑运算指令可以实现二维像素级并行操作的效果，但是我们也需要考虑使用这类指令完成计算的周期数、以及涉及的存储空间。如果使用按位操作进行乘法计算，则将消耗大量的时钟周期，这样相当于抵消了二维并行带来的速度优势，因此复杂的操作应该由 Lane 的算术运算指令或者 PPU 来完成。

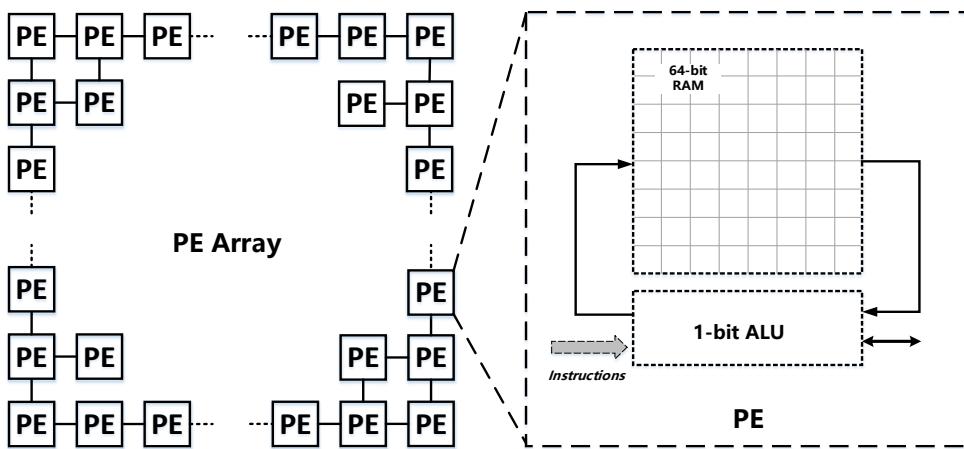


图 3.17 传统 PE 电路示意图

本文所设计的 PE 电路区别于传统的 PE 设计。如图 3.17 所示，传统 PE 阵列中数据是被分布的存储在每个 PE 的存储器中的，而通用处理器中，数据却集总的存储在一个或者少数几个存储器中。分布式的存储器方式使 PE 阵列具有极高的数据吞吐率，以图 3.17 中的 PE 阵列为例，若其阵列大小为 64×64 ，每个 PE 存储器的大小为 64 比特，每次 PE 中的 ALU 可以访问到其存储器中 1 比特的数据。在一个 SIMD 数据访问指令的驱动下，PE 阵列可以得到 $64 \times 64 = 4096$ 比特的数据，并且整个 PE 阵列在一个时钟周期下也可以完成 4096 比特的运算。然而，对一个 32 比特通用处理器而言，每个时钟周期仅仅可以通过数据存取指令获得 32 比特数据或者通过数据处理指令处理 32 比特数据，据吞吐率上相差 $4096/32=128$ 倍。由此可见，2 维 PE 阵列在性能上的确具有较大的优势，但是如此巨大的吞吐率带来的问题是引入过多细粒度的存储器。实现图 3.17 所示的 PE 阵列需要 4096 个存储器，并且每个存储器容量非常小，仅为 64 比特。众多存储器造成系统存储器面积过大，并且单个存储空间小，无法被其他处理器所利用。以目前视觉芯片设计为例，其 PE 存储器占整个系统存储容量超过

70%^[31, 48]，然而，在实际的视觉或者图像处理过程中，PE 阵列的使用率并不高。如图 3.18 所示，为在视觉芯片[48]上完成目标追踪算法时各个处理器所消耗的时间，PE 阵列在占整个系统超过 80% 以上硬件资源的情况下只占整个处理时间的 13%。以上述 PE 阵列为例，PE 存储器容量为 $64 \times 64 \times 64$ 比特—32KB—仅仅在 PE 阵列工作的时候是有效用的，其他 87% 的时间这 32KB 的存储容量处于闲置状态。通过上述的电路设计方式，PE 电路不在单独使用存储器，并且控制逻辑也由 Lane 的控制电路完成，减少了硬件资源的消耗。

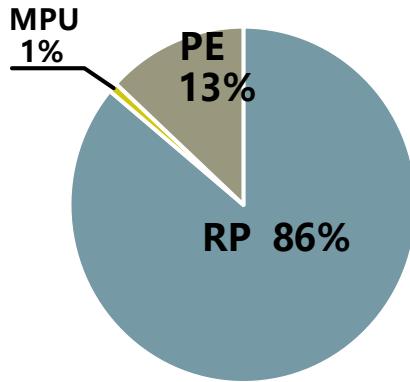


图 3.18 视觉芯片[48]完成追踪算法时各处理器耗时

3.5 缓存电路设计

在 HERO 架构中存在多个主处理器访问程序存储器的情况，如果只使用单一存储器，势必会造成冲突。所以在每个可取指的 PPU 处理器中设计了一级缓存(Cache)，以减小不同 PPU 取指令时在指令存储器处出现的冲突。

Cache 的设计概念主要来自与程序的空间局部性（Spatial Locality）和时间局部性（Temporal Locality）^[52]。程序的空间局部性是指某一段程序在某一时刻被执行之后，其附近的程序段有很高的几率在后续的执行中被访问。程序的时间局部性是指一段程序在某一时刻被执行之后，往往在短暂停时间之后将有极大的可能被再执行一次。如图 3.19 所示，Cache 为一小块能被处理器快速访问的存储器，其中存储了处理器近期访问的程序段以及该程序段附近的程序段。在处理器需要访问数据或者指令时，其首先访问 Cache，如果 Cache 中存储了该数据和指令，那么处理器可以立即获取这一数据或指令。如果 Cache 中没有存储该数据或者指令，那么处理器再从主存中获取需要的数据和指令。考虑到程序的空间局部性和时间局部性，大部分的数据或指令都能在 Cache 中立即获取到，而减少了处理器从

主存中获取数据、指令所消耗的时间，提高了系统的性能。

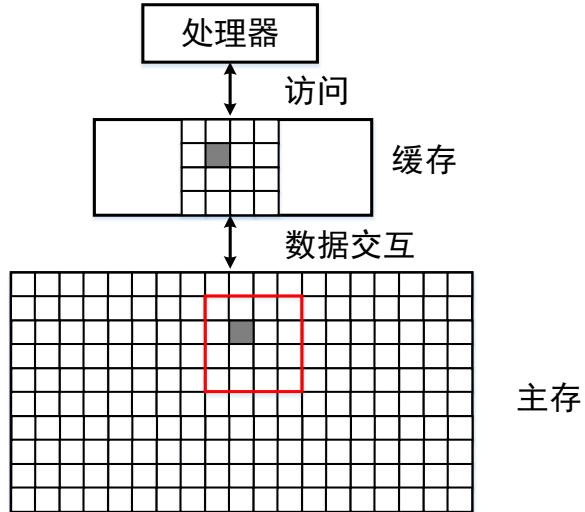


图 3.19 Cache 示意图

在 HERO 架构中，PPU 中增加了一级指令 Cache，以便 PPU 大部分需要获取的指令能够在 Cache 中得到，这大大减少其访问主程序存储器的概率，也就是减少了多个 PPU 访问主程序存储器所产生的冲突，提高了处理速度。我们首先考虑几种常见的 Cache 设计的方法，分别是 Direct-mapped，Set-associative 以及 Full-associative。如图 3.20 所示，假设主存储器的数据被分为了 31 个 block（每个 Block 包含数个字节），而缓存的大小可以存储 8 个 block 的数据。对于 Direct-mapped 这种形式的缓存，主存中某一位位置 Block 的数据只能映射到缓存中的某一固定位置，该位置由式子（3-5）计算得到：

$$(\text{Block 地址}) \bmod (\text{Cache 中 Block 数量}) \quad (3-5)$$

那么图 3.20 中主存中紫色标记的 Block No.12 在 Direct-mapped 的缓存中只能映射到缓存中 Block No.4 ($12 \bmod 8 = 4$) 的位置。如果 Cache 中 Block 数量是 2 的幂次，那么计算式（3-5）时，只需要取相应地址二进制码的低位即可。在图 3.20 中， 12_{ten} (01100_{two}) 对应缓存中的地址为 100_{two} ，也就是地址的后 3 位 ($\log_2 8$)。我们很容易发现，主存中位置在 $4_{\text{ten}}(00100_{\text{two}})$ 的数据也应该被映射到缓存 Block No.4 的位置，也就是当处理器需要访问主存中 Block No.4 中的数据时，缓存中的数据就会被替换掉。解决这一问题的方式是采用 Set-Associative 缓存，这类缓存允许主存中的某一 Block 中数据被映射到缓存中的 N 个不同位置，这样的缓存被称作 N-way Set-associative 缓存。对于图 3.20 中的 2-way Set-Associative 缓存，主存中 Block

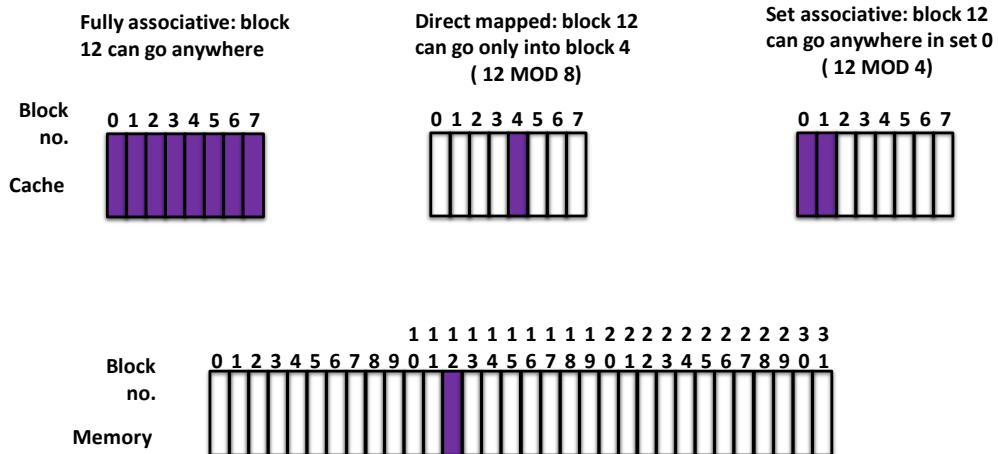


图 3.20 几种常见的 Cache

No.12 中的数据，可以映射到缓存中 Block No.0 和 No.1 的位置，其计算方法为：

$$(\text{Block 地址}) \bmod (\text{Cache 中 Sets 数量}) \quad (3-6)$$

其中，缓存中组 (Set) 的数目由其 Block 数目以及 N 来共同。比如在 Cache 中有 8 个 Block，N 为 2，那么 Set 为 4 ($8/2=4$)。Full-associative 缓存中，主存中任意 Block 位置的数据可以被映射到缓存中任意 Block 位置。

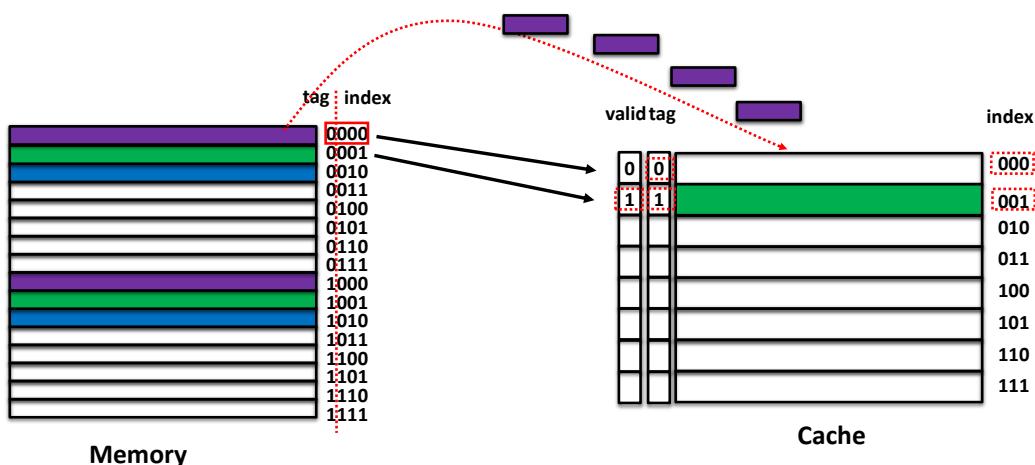


图 3.21 主存地址映射到缓存

上述三种 Cache 设计中，Direct-Map 缓存设计最为简单，为了方便实现并且减少硬件资源消耗，我们使用其作为本文的 Cache 实现方式。为了记录 Cache 与主存之间正确的映射关系，存储器中 Block 地址被划分为两部分，Tag 和 Index。图 3.21 中，主存 Block 地址从 0_{ten} (0000_{two}) 至 15_{ten} (1111_{two})，其中地址低三位作为 Index，高位地址作为 Tag。当数据需要

从主存映射到 Cache 中的时候，根据数据在主存中的 Index 直接映射到相应的 Cache Block 中，并且在 Cache 中标记其 Tag 位。Cache 中 Valid 位置表示该 Block 的数据是否有效，完成主存到缓存之间的映射后需要将该位置为 1，以表明其有效。图 3.21 中，缓存 Index 为 000 的 Block，因为 Valid 位为 0，所以数据是无效的。Index 为 001 的位置可以对应了主存中 0001 或者 1001 地址的数据，但是因为 Tag 位为 1，所以该 Block 对应的是主存中 1001 地址的数据。

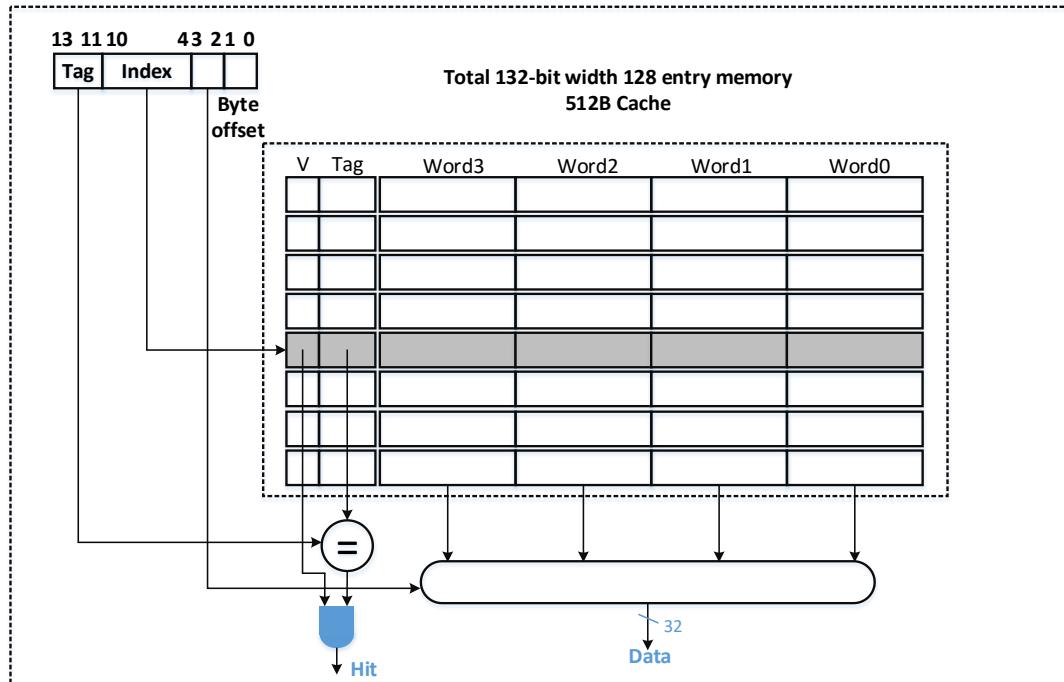


图 3.22 Cache 实现电路图

图 3.22 为本文 Direct Mapped 缓存电路实现。该 Cache 中每个 Block 包含 4 个字，Cache 大小为 512 字节，也就是 $512/4=128$ 个 Block。在接收到处理器发送过来的地址数据后，首先通过 Index（地址第 4 至 10 比特）选择 128 个 Block 中的一个 Block，并且取得该 Block 对应的 Tag 值和 Valid 值。如果 Tag 值和地址高位（11 至 13 比特）相同并且 Valid 等于 1，那么该数据为有效数据，处理器将取走该数据。如果 Tag 值和地址高位不相等或者 Valid 位不等于 1，那么我们就要借助缓存控制器从主存中取得正确的数据。图 3.23 为本文所设计缓存控制器的状态机。该状态机总共有空闲（Idle），比较 Tag（Compare Tag）以及定位（Allocate）三个状态。当处理器向缓存请求数据时，控制器进入 Compare Tag 状态，该状态进行 Tag 比较，如果数据存在则被处理器读走，状态机恢复到 IDLE 状态并等待处

理器下一次请求。如果 Compare Tag 失败，则控制器进入 Allocate 状态，该状态下控制器将获取正确的主存数据并更新控制器中 Tag 位，该状态完成后状态重新进行 Compare Tag 状态。

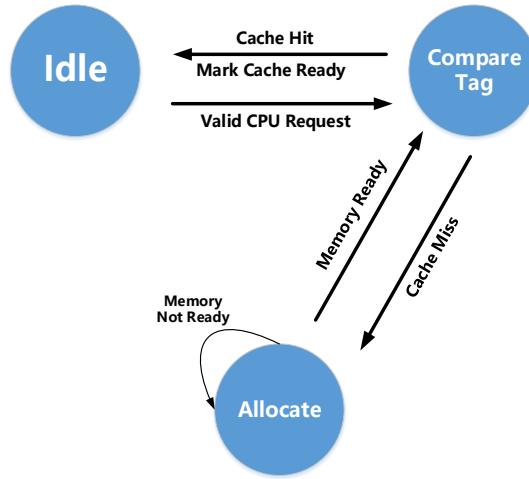


图 3.23 Cache 状态机

3.6 本章小节

本章介绍了 HERO 架构中的指令集和关键电路设计，包括 PPU 电路设计，Lane 电路设计，PE 电路以及缓存电路设计。本章详细介绍了 Lane 电路进行分布式并行处理以及快速直方图统计的方法，PE 阵列在像素级并行处理的情况下多比特运算的实现，并阐述了 PE 电路和 Lane 电路之间的关系。

第4章 特征检测与描述算法

本文前面几章节的内容介绍了视觉芯片架构，而面向架构的算法设计则是实现其价值的关键环节。以往的视觉芯片算法主要集中在滤波、边缘检测、运动检测以及形态学运算等较为初级的图像处理方面，为了更好的适应现代视觉信息处理的框架，本章将介绍特征点检测、特征描述算法，并研究如何在视觉芯片上并行实现这些算法。

4.1 特征点检测

特征点检测在图像中选择出具有代表意义的一些点或者区域，这些点或区域可以用来帮助我们进行识别图像、进行图像匹配等工作。特征点是指图像上灰度相对于所在区域有剧烈变化的像素点，比如线段的终点、曲线中曲率最大的点等等，因此特征点具有一定的旋转、光照甚至遮挡的不变性，这些特征点是图像的重要特征。

4.1.1 最小同值分割吸收核（SUSAN）角点检测

如图 4.1 所示，为一个长方形的图像，图像的上部区域较亮，下部区域较暗，分别可以认为是目标和背景。图中用“+”所标记的像素点称为“核”，以此为圆心，半径相同的六个圆形模板落在了不同的区域。如果我们比较这几个模板之中各个像素和中心核像素的灰度，那么我们总会发现有一部分模板像素的灰度与核像素的灰度相同或者相似。这部分区域可被称为

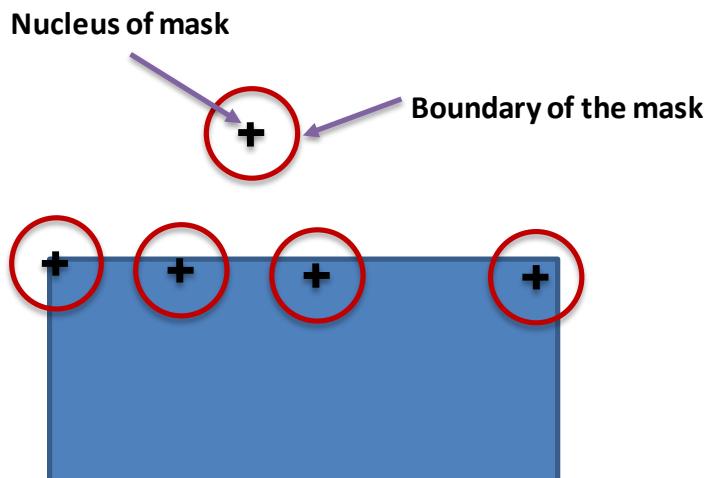


图 4.1 SUSAN 角点检测原理

USAN (Univalue Segment Assimilating Nucleus) 区域。SUSAN (Small Univalue Segment Assimilating Nucleus) 角点就是利用 USAN 区域的面积大小来判断核像素点是否为角点^[53]。

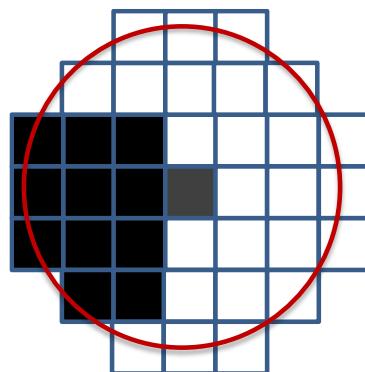
SUSAN 算子采用圆形模板来得到各项同性的响应。我们可以使用图 4.2 中含有 37 个像素的圆形模板，该模板总共七行，分别有 3,5,7,7,7,5,3 个像素。这相当于半径为 3.4 个像素的圆。为了计算每个模板的 USAN 值，我们用式 (4-1) 来进行计算：

$$C(x_0, y_0; x, y) = \begin{cases} 1 & |f(x_0, y_0) - f(x, y)| \leq T \\ 0 & |f(x_0, y_0) - f(x, y)| > T \end{cases} \quad (4-1)$$

式中， (x_0, y_0) 是核在图像中的位置， (x, y) 是模板中其他像素的位置。 $f(x_0, y_0)$ 和 $f(x, y)$ 分别是在 (x_0, y_0) 以及 (x, y) 处像素的灰度值。T 为一个人为设定的阈值。对模板中的每个像素进行上述比较，然后进行累加求和可以得到：

$$S(x_0, y_0) = \sum_{(x, y) \in N(x_0, y_0)} C(x_0, y_0; x, y) \quad (4-2)$$

最后的结果给出了 USAN 区域的面积大小，我们进一步可以使用该值与阈值比较大小以判断中心核是否为角点。当该值小于某个阈值时，我们认为



37 pixel mask

图 4.2 本文用于 SUSAN 角点检测的模板

为模板中间的像素为角点，反之认为其不是角点。阈值一般选取为 $3S_{\max}/4$ ，其中 S_{\max} 是式 (4-2) 中 S 所能取得的最大值。

4.1.2 Fast 特征点检测算子

近年来，一种叫做 FAST 的算子，因其简单高效的计算以及良好的实验性能而脱颖而出^[54]。FAST 算子如图 4.3 所示，对灰度图像中任意一个像素 P ，与环绕它的临近 16 个像素进行灰度值比较，如果这 16 个像素组成的

环形中存在连续 9 个更亮或者更暗的像素，则认为 P 像素为特征点。如式 (4-3)，对任意 $x \in \{0 \dots 15\}$ ，用 $I_{p \rightarrow x}$ 表示该像素的灰度值，如果 $I_{p \rightarrow x}$ 小于中心像素灰度值 I_p 与阈值 t 之差，那么认为该像素比中心像素更暗 (Darker)，如果 $I_{p \rightarrow x}$ 大于中心像素灰度值 I_p 与阈值 t 之和，那么认为该像素比中心像素更亮 (Brighter)，如果 $I_{p \rightarrow x}$ 处于上述两者之间，那么认为该像素和中心像素的灰度值相近 (Similar)。

$$S_{p \rightarrow x} = \begin{cases} d, & I_{p \rightarrow x} \leq I_p - t \quad (\text{darker}) \\ s, & I_p - t < I_{p \rightarrow x} < I_p + t \quad (\text{similar}) \\ b, & I_p + t \leq I_{p \rightarrow x} \quad (\text{brighter}) \end{cases} \quad (4-3)$$

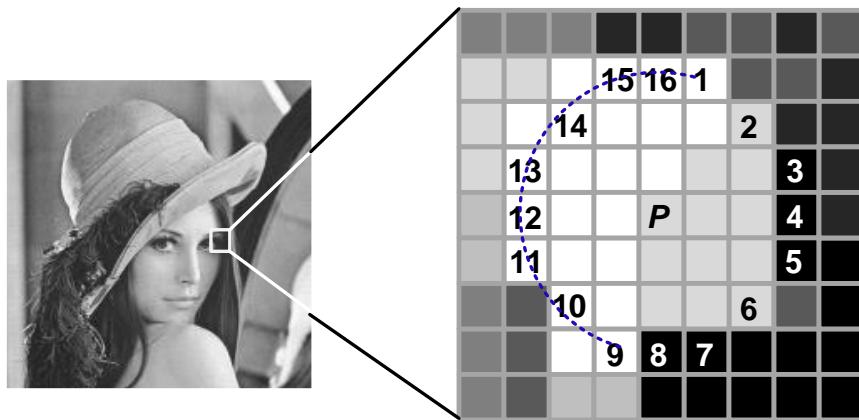


图 4.3 FAST 特征点检测算法

经过灰度值比较以后就需要确定是否存在连续暗 (亮) 像素，这一过程被称为段测试 (Segment Test)。文献[55]使用决策树方法，使得每个像素对应的段测试能够很快完成。然而，决策树的方法涉及到过多的条件控制和跳转，在 PE 阵列中并不适用。我们可以使用穷举测试 (Exhaustive Test) 的方式来完成段测试，也就是对这 16 个像素都进行一遍测试，每次测试检测从该点开始是否存在连续 9 个像素更亮或者更暗。具体的：首先对于任意一个像素 $x \in \{0 \dots 15\}$ 利用式 (4-4) 与中心像素进行灰度值比较并得到两个标志位 b_i 和 d_i ，分别表示其是否比中心像素更亮或者更暗。最后，通过式 (4-6) 完成穷举测试。

$$b_i = s((I_p + t) - I_{p \rightarrow x}), \quad d_i = s(I_{p \rightarrow x} - (I_p - t)) \quad (4-4)$$

其中，

$$s(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \quad (4-5)$$

$$B = \sum_{i=0}^{15} \prod_{j=0}^8 U(b_{i,j}), \quad D = \sum_{i=0}^{15} \prod_{j=0}^8 U(d_{i,j}) \quad (4-6)$$

其中，

$$u(x_{i,j}) = \begin{cases} x_{i+j-15}, & \text{if } (i+j) > 15 \\ x_{i+j}, & \text{otherwise} \end{cases} \quad (4-7)$$

完成穷举测试后，我们还可以对每个测试点完成非极大值抑制（Non-Maximal Suppression）操作：

$$V = \max(V_b, V_d) \quad (4-8)$$

其中，

$$V_b = \sum_{i=0}^{15} b_i * (|I_{p \rightarrow x} - I_p| - t), \quad V_d = \sum_{i=0}^{15} d_i * (|I_p - I_{p \rightarrow x}| - t) \quad (4-9)$$

在上述方法中，由于 b_i 和 d_i 为 1-bit 变量，式 (4-6) 以及 (4-9) 当中的乘法运算都可以利用逻辑与运算来完成。并且，由于任何两个像素的计算没有任何相关性，所以上述过程是完全可以像素并行完成的。该特征点检测算法在视觉芯片下的高速实验结果将在后续章节中给出。

4.1.3 尺度不变特征点检测

尺度不变（Scale Invariant）特征点检测算法是最为成功的特征点检测算法之一，在很长一段时间内受到了极高的关注度，并且广泛应用于图像匹配、识别和三维重建等视觉领域^[56]。

要了解尺度不变，我们必须要先了解尺度空间（Scale Space）^[57]的概念。这一概念在计算机视觉中尤为重要，我们在此不妨做简要介绍。我们知道现实世界中的事物是具有尺度概念的，比如说当描述树的时候我们显然应该使用“米”这一单位，而考虑一片叶子的时候，我们应该使用更小的单位（比如“厘米”）。然而，当一个视觉系统分析未知场景的时候却是无法获得任何关于描述其中物体尺度的先验知识。我们考虑图 4.4，虽然两张图片展示的都是同一事物，但是由于取景深度不同，导致两图的尺度相差较大。如果我们有一个特征函数（Signature Function）来衡量两图中同一位置为中心某范围内的响应。那么我们可以得到图 4.4 中的两个响应曲

线，其中横轴即为范围尺度，纵轴为响应。因为两个响应曲线分别对应的是两张图中同一点，那么它们的响应曲线形状应该相对相似，唯一区别是其中一个响应曲线被“压缩”或者“拉升”。不难想象，两个曲线的最大值应该对应相同的范围，因此如果它们对应的尺度分别是 σ 和 σ' ，那么两个图的尺度之比就是 σ/σ' ，而上述求解最大响应的这一过程就构成了尺度空间。

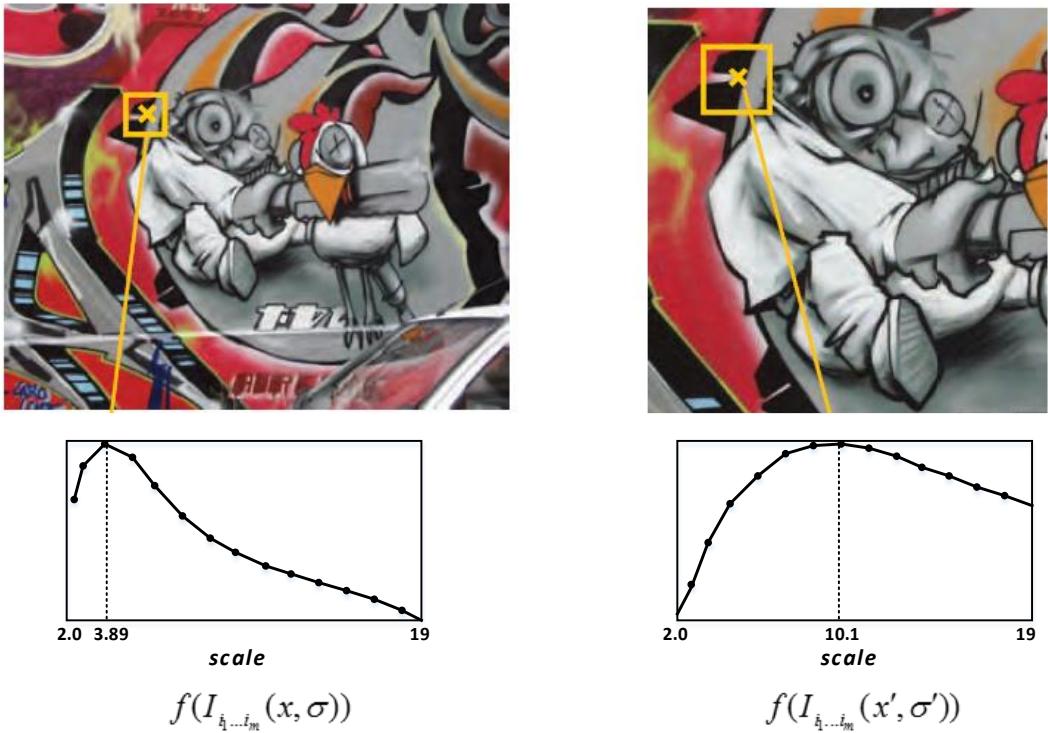


图 4.4 给定一对特征点，对特征点周围区域使用不同尺度，并求特征函数响应。因为这一对特征点对应图像中的相同位置，所以它们的响应函数应该相似，并且可以通过寻找响应极大值得到响应的邻域尺度。

1994 年 Lindeberg 证明了能够唯一作为特征函数的是归一化的高斯核^[58, 59]。具体的，一个图像的尺度空间 $L(x, y, \sigma)$ ，定义为原始图像 $I(x, y)$ 与一个可变尺度的 2 维高斯函数 $G(x, y, \sigma)$ 的卷积运算。在此基础上，他提出了用尺度归一化的拉普拉斯-高斯 (Laplacian-of-Gaussian) 算子用于搜索尺度空间极值^[59]的方法。Lowe 进一步于 2004 年提出该算子可以用高斯差分 (Difference-of-Gaussian, DoG) 来近似，并且在计算上更加简便，具体的：

$$D(x, \sigma) = (G(x, k\sigma) - G(x, \sigma)) \otimes I(x) \quad (4-10)$$

其中， $G(x, \sigma)$ 为高斯核。Lowe 证明，当系数 K 为常数时，上式自动包含了所需的尺度归一化要求^[5]。通过 DoG 算子，也就是高斯差分方法，我们

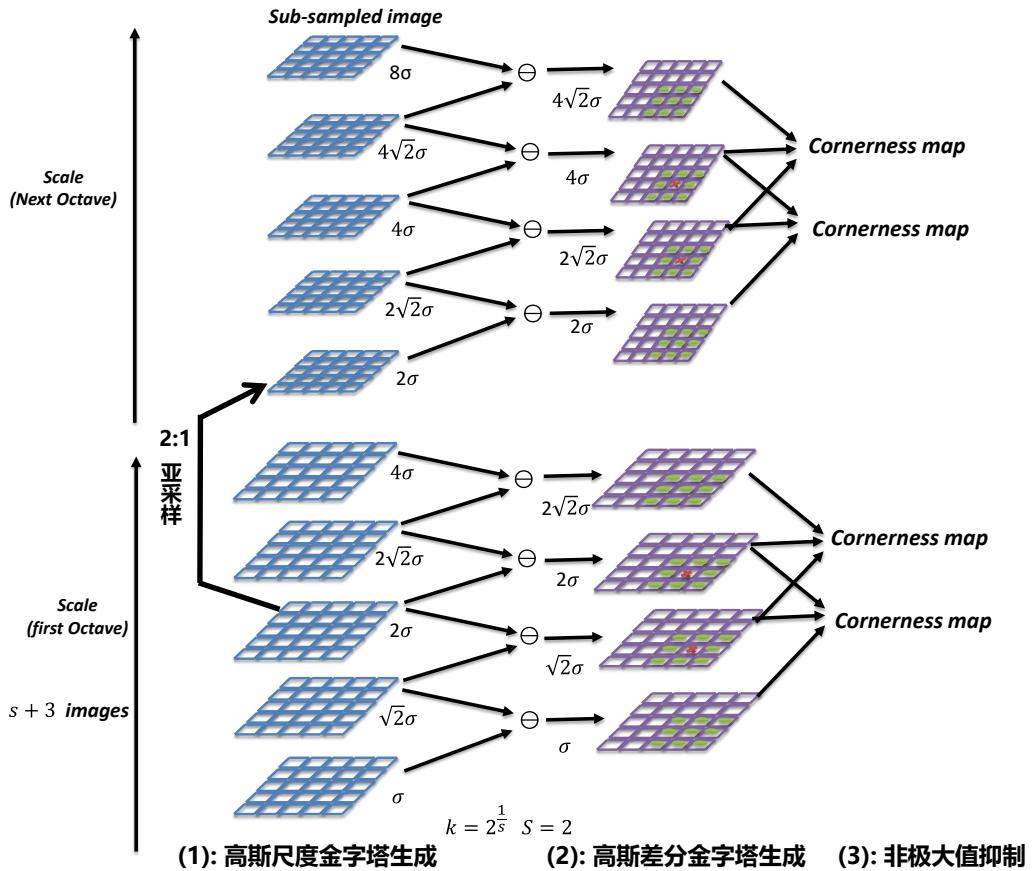


图 4.5 SIFT 尺度不变特征点检测算法

能够找到图像在不同尺度上的响应极值。如图 4.5 所示，为 Lowe 提出的算法包括三个步骤^[5]：

- (1) 利用不同的高斯卷积核对图像进行滤波得到高斯金字塔。整个高斯金字塔包含 O 个组（Octave），其中每个组包含了 S 个尺度（Scale）。两个相邻尺度的比例因子（式 (4-10) 中 K ）一般选择为 $\sqrt{2}$ ，每组一般包含 2 个尺度。因此，如图 4.5 所示，第一组中高斯核 σ 的取值为 $\sigma, \sqrt{2}\sigma, 2\sigma \dots$ 。将第一组中倒数第三幅高斯图像进行水平和垂直方向 2:1 采样即可得到第二组的第一幅高斯尺度，再通过相同的高斯滤波函数对这幅图像进行滤波就可得到第二组高斯尺度空间。对图像进行亚采样相当于在对滤波贡献了一个尺度为 2 的因子，因此第二组的尺度分别为 $2\sigma, 2\sqrt{2}\sigma, 4\sigma \dots$ 。这样做的目的显而易见，随着尺度因子的增大，计算量变得越来越大，而通过降采样的方法，达到了所需的尺度因子，且计算量并不增加。

- (2) 利用上一步生成的高斯金字塔做差生成高斯差分金字塔，那么第一组和第二组就分别求得了 4 个尺度的 DoG 响应。
- (3) 非极大值抑制的过程就是求得最值的过程。我们考虑图 4.4 中的曲线，极大值响应高于小于其尺度的响应，同时也高于大于其尺度的响应，对应到图 4.5 也就是在纵轴上响应高于其上尺度与其下尺度的点。通过非极大值抑制，我们可以求得 $\sqrt{2}\sigma$, 2σ , $2\sqrt{2}\sigma$, 4σ 这四个尺度下的最值。其中 $\sqrt{2}\sigma$ 和 2σ 通过第一组求得，而 $2\sqrt{2}\sigma$, 4σ 通过第二组求得，两组之间任然呈现 $\sqrt{2}$ 的倍数关系。这也是为何第二组要从 2σ 尺度处开始的原因：保证两组之间连续性。

事实上，图 4.5 的纵轴与图 4.4 的横轴表示的意义是相同的，都是图像在不同尺度下的响应，最后非极大值抑制的目的就是求出响应极值。虽然 Lowe 所提出的尺度不变角点检测算法广泛应用，但是其弱点在于多次的浮点型高斯滤波导致运算量很大，实现起来速度比较慢。使用 ARM A8 处理器对 1024×768 大小图像完成完整的 SIFT 算法耗时达 53 秒，即便是在

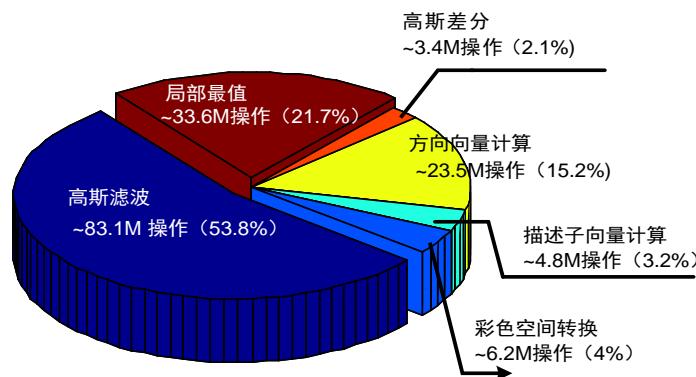


图 4.6 SIFT 算法中各个步骤所需操作分布

Inter Core 2 Quad 处理器上也需要 1.6 秒^[60]。SIFT 算法中各种运算所消耗的运算量如图 4.6 所示，其中超过 50% 的运算都用于进行高斯滤波^[61]。由此可见，实现高速尺度不变特征点提取的关键在于快速、高效的实现不同的高斯滤波核。

我们利用高效高斯函数的叠加性（Cascading Gaussian）和可分性（Gaussian Separability）来高效构建高斯滤波核^[62]。高斯叠加性是指，高斯函数 $G(x, y, \sigma_1)$ 与 $G(x, y, \sigma_2)$ 的卷积等效于 $G(x, y, \sqrt{\sigma_1^2 + \sigma_2^2})$ ，这也就意味着如果我们需要对图像进行标准差为 $\sqrt{2}\sigma$ 的高斯滤波，那么我们可以使用连续两次标准差为 σ 的高斯滤波来等效。进一步的，如图 4.7 所示，如果

我们对一副图像使用 $G(x, y, \sigma)$ 连续滤波 n 次，那么第 n 次的滤波结果就相当于使用 $G(x, y, \sqrt{n}\sigma)$ 对图像滤波的结果。通过这种方法，如果我们取第 1、2、4、8、16 次连续滤波结果，就可以得到图 4.5 中的 σ , $\sqrt{2}\sigma$, 2σ , $2\sqrt{2}\sigma$, 4σ 标准差滤波的结果。

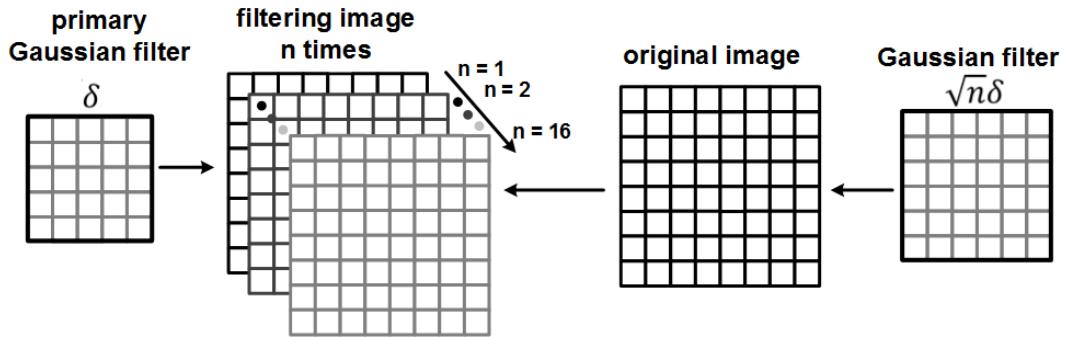


图 4.7 利用多次高斯滤波来构建尺度空间

利用高斯可分性，如果上述二维高斯函数 $G(x, y, \sigma)$ 可分，我们可以进一步的将其拆分为两个一维高斯函数^[62]。图 4.8 所示展示了我们选用的几个一维高斯函数，其 σ 值正好呈 2 倍关系。如果我们使用 $\sigma = 0.5$ 的模板分别进行垂直和水平方向的滤波，那么就相当于进行了一次 $\sigma = 0.5$ 的二维滤波。重复上述滤波 16 次，就可以得到第一组高斯尺度图像。如果我们使用 $\sigma = 1.04$ 的模板也进行垂直、水平滤波，并重复 16 次，那么就可以得到第二组高斯尺度图像（这里不使用亚采样，因为因子 2 已经使用包含在了更大的高斯模板中）。图 4.8 所示几个高斯模板很容易使用移位方式实现而消除了乘法、浮点运算，并且滤波操作在视觉芯片中可以高速实现，所以使用上述方法来完成尺度不变角点将大大提高性能，其实现性能将在后续章节给出。

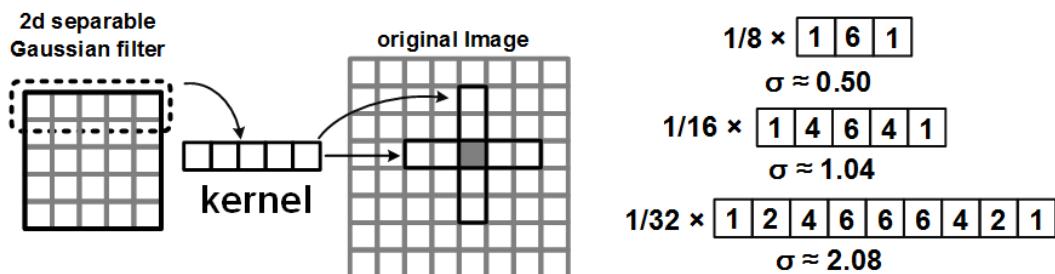


图 4.8 利用两个 1 维高斯滤波来等效 2 维高斯滤波

4.2 特征描述

4.2.1 LBP 特征描述

局部二值模式 (Local Binary Patten, LBP) 局部特征描述算法是对图像局部纹理特征进行描述的方法，因其具有旋转不变性和灰度不变性等显著的优点，而被广泛应用在人脸检测、人脸识别领域^[41, 43, 63]。图 4.9 为一个基本的 LBP 算子，在任意 3×3 临域内，以中心像素为阈值与半径为一个像素所对应圆环下八个像素的灰度值进行比较，若周围像素大于中心像素，则该点被标记为 1，反之则被标记为 0，计算所得为 8 个 0 或者 1 组成的字符串。大量的实验数据证明，超过 90% 的纹理模式所得到的字符串中 $0\rightarrow 1$ 或 $1\rightarrow 0$ 转换不会超过两次（如图 4.10 的第 1、2 行所示），只有小于 10% 的纹理模式得到的字符串中 $0\rightarrow 1$ 或 $1\rightarrow 0$ 转换则会超过两次（如图 4.10 的第 3 行所示）^[41]。我们将具有不超过两次 $0\rightarrow 1$ 或者 $1\rightarrow 0$ 转换的纹理模式称为统一模式 (Uniform Pattern)，而剩余的模式则称为非统一模式 (Non-uniform Pattern)。根据统一模式中包含 1 的数量，可以对该种模式进行编码 (0...8)，而非统一模式则统一编码为 9。

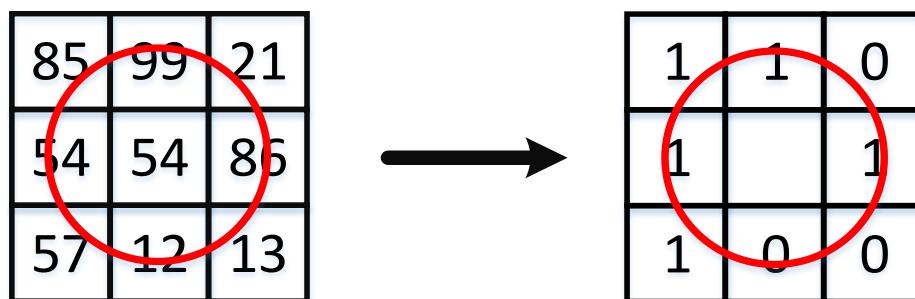


图 4.9 基本 LBP 算子，将中心像素作为阈值与周围八个临近像素进行比较得到由 8 个 0 或者 1 组成的字符串。

我们可以用式 (4-11) 对上述基本 LBP 算子进行数学描述：

$$LBP_{P,R} = \begin{cases} \sum_{p=0}^{P-1} s(g_p - g_c) & \text{if } U(LBP_{P,R} \leq 2) \\ P+1 & \end{cases} \quad (4-11)$$

其中，

$$s(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \quad (4-12)$$

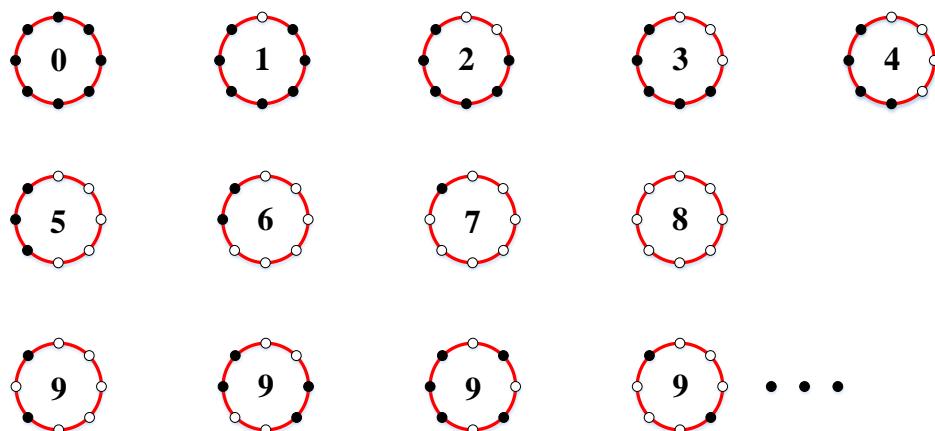


图 4.10 基本 LBP 算子可以产生的字符串，其中黑色点表示 0，白色点表示 1。第一、二行是统一模式，而第三行则为非统一模式。

$$U(LBP_{P,R}) = |s(g_{p-1} - g_c) - s(g_0 - g_c)| + \sum_{p=1}^{P-1} |s(g_p - g_c) - s(g_{p-1} - g_c)| \quad (4-13)$$

在式 (4-11) 与 (4-13) 中，R 表示中心像素到临近像素的半径，而 P 表示了临近像素的个数，R 和 P 有多种取值方式，比如 (P=12, R=1.5)，(P=16, R=2)，(P=24, R=3) 等。为了减少计算量同时消除不必要的亚像素运算，我们往往选择 (P=8, R=1) 和 (P=16, R=2)。在图 4.9 所展示的实例中，P=8, R=1。

经过上述 LBP 算子，我们已经将图像转化为 LBP 图，LBP 图中每个像素的 LBP 值为 $x \in \{0 \dots P + 1\}$ ，后续的局部特征就是在该图像上进行直方图运算得到的。如图 4.11 所示，将 LBP 图像划分为 $M \times M$ 个图像块，对每一个区域所求得的 $P+2$ 维直方图则为该子块对应的局部特征向量。将这些特征级联起来，即可得到整个区域的特征向量，一个 $M \times M \times (P + 2)$ 维度的直方图。

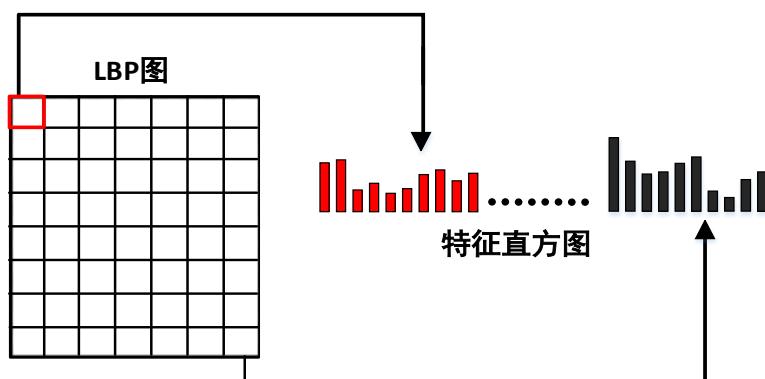


图 4.11 利用 LBP 图构建特征向量

从上面的分析可以看出利用 LBP 来构建特征向量分为两步：第一步，对原始图像施加 LBP 算子，得到 LBP 图；第二步，将 LBP 图分割为若干子块，并对每个子块进行直方图统计，得到局部特征向量。局部特征向量构建完成之后，整个图像的特征向量也就构建完成了。第一步的运算只涉及到逻辑以及加减法运算，并且对任意两个像素而言，其 LBP 算子的计算是没有任何依赖性的，因此 LBP 图像生成过程是可以像素并行的。具体的，我们可以用式（4-14）和（4-15）表示：

$$f(x) = \begin{cases} 9, & \text{if } (\sum xor_i > 2) \\ \sum_{i=0}^7 b_i, & \text{else} \end{cases} \quad (4-14)$$

$$xor_i = \begin{cases} b_i \wedge b_{i+1}, & 0 \leq i < 7 \\ b_i \wedge b_{i-7}, & i = 7 \end{cases} \quad (4-15)$$

在第二步中，子块与子块之间的运算是没有依赖性的，因此这一步是可以块并行的。LBP 特征描述在视觉芯片下的实现结果以及性能将会在第六章中给出。

4.2.2 HoG 特征描述

方向梯度直方图（Histograms of Oriented Gradients, HOG）是 Dalal 等人于 2005 年提出的一种局部特征描述方法，由于其对图像优良的表达性能，而被广泛的使用于行人、人脸以及姿态识别当中^[42, 64, 65]。HoG 描述算法的第一步是计算图像中每个像素在水平和垂直方向上的梯度：

$$\frac{\partial f}{\partial x} = f(x+1) - f(x), \quad \frac{\partial f}{\partial y} = f(y+1) - f(y) \quad (4-16)$$

在图像处理中，我们可以 $[-1 \ 0 \ 1]$ 以及 $[-1 \ 0 \ 1]^T$ 两个滤波器来分别实现两个方向的梯度计算。那么任意像素点 (x, y) 对应的梯度幅值 $M(x, y)$ 以及方向 $\theta(x, y)$ 可分别表示为：

$$M(x, y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \quad (4-17)$$

$$\theta(x, y) = \arctan\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \quad (4-18)$$

可以看出，梯度幅值和方向的计算开销是比较大的，如果完全使用硬件实现引入浮点计算，则会增加不少硬件面积。我们可以使用如下方法对梯度幅值进行近似：

$$M(x, y) \approx \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right| \quad (4-19)$$

同时，实验结果显示，仅仅区别 $0^\circ \sim 360^\circ$ 之间的八个方向就可以达到对图像的表达效果^[42]。因此，我们只需要考虑像素点的梯度方向具体落在 $0^\circ \sim 45^\circ$, $45^\circ \sim 90^\circ, \dots, 315^\circ \sim 360^\circ$ 这八个区间哪一个即可。对任意像素点计算所得的两个梯度 (dx, dy)，只需要对其做三个边界检测即可得到其梯度方向落在哪个区间之中。比如，当 $dx > 0, dy > 0, |dx| > |dy|$ 三个条件都满足时，该点的梯度方向落到 $0^\circ \sim 45^\circ$ 区间。在完成了上述运算后，使类似用图 4.11 所示的方式，将图像分为若干块，并完成每个图像块中的梯度幅值与方向的统计并级联得到特征向量。

4.2.3 其他特征描述方法

SIFT 特征描述方法与 HoG 有相似之处，也是基于图像中梯度的方向和模进行统计的，只是在进行最后的特征向量构建方法上有所不同。我们简化 SIFT 算法特征向量的构建方法如图 4.12 所示，图(a)中央白点为用检测得到的特征点，每个小格代表特征点邻域所在尺度空间的一个像素，邻域范围共包含 8×8 个像素。箭头方向代表该像素的梯度方向，长度则代表了梯度强度，梯度方向和幅值的计算，可以继续利用上一小节中的方法。在每 4×4 个像素的更小图像块上计算 8 个方向的梯度直方图，如图(b)所示。最后将得到的四个直方图级联得到图(c)所示的特征向量。图 4.12 中特征向量的构成过程虽然没有主方向确定和旋转，但是根据 Lowe 的研究，

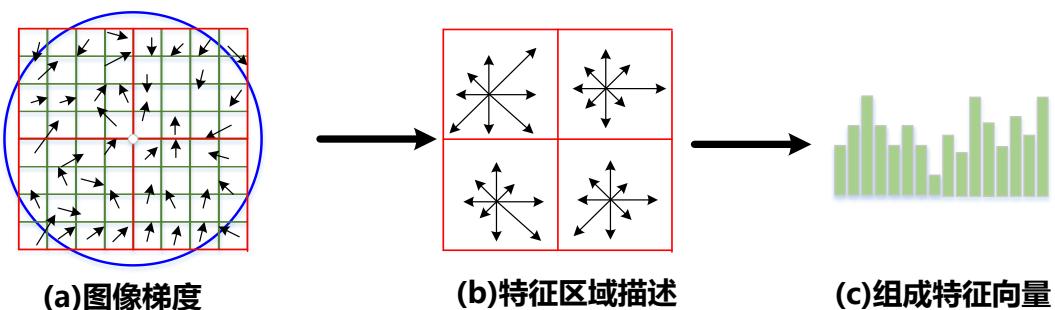


图 4.12 SIFT 特征向量构造

这样特征描述构建方法所产生的特征足够识别旋转在 $+15^\circ$ 至 -15° 范围内的物体。

SURF 算法进行局部特征描述时，首先选定特征点为中心的一块正方形区域，并将其正方形区域划分为 4×4 的 16 个子块，对每个块进行 $\sum dx$ 、 $\sum |dx|$ 、 $\sum dy$ 、 $\sum |dy|$ 四个系数的计算，从而生成 $4 \times 4 \times 4$ 共 64 维特征向量。

通过上面的分析，我们看到 SIFT、SURF 特征向量构建的主要过程涉及到小图像块中像素梯度、幅度的计算以及直方图的统计，这些运算是可以在 HERO 中并行完成的。但是，SIFT 和 SURF 算法中每个特征点都会产生一个特征向量，每个特征向量的存储需要 256 字节、512 字节或者更大的存储空间，而一幅图像中检测到的特征点数目一般在 1000 以上的，因此在进行计算时，需要占用的存储空间很大，对于视觉芯片这样的片上系统芯片设计，在实际运用中更适合使用 LBP、HoG 一类特征描述方法。

4.3 本章小节

本章重点研究了用于视觉芯片的特征检测和特征描述算法。介绍了 SUSAN、FAST、SIFT 特征检测算法的原理并提出了它们像素级并行实现的方法，介绍了 LBP、HoG、SIFT、SURF 几种广泛使用的局部特征描述方法，并对算法进行部分改进以适应可能的视觉芯片片上处理。

第5章 面向视觉芯片的并行分类算法

特征分类是指在构建出目标特征后，利用目标特征对目标进行分类的过程，该过程主要使用模式识别算法来完成。本章将介绍在图像识别领域普遍使用，同时也取得较好效果的 SVM, SOM 神经网络和 AdaBoost 算法，并深入分析这三种算法的本质，并分析其在视觉芯片上并行实现的可能性和方法。

5.1 图像特征识别

人类可以通过“一瞥”即可对眼前所见的事物进行准确无误的分类，在日常生活中，我们无时无刻不在进行着各种各样的分类。模式识别作为一门 20 世纪兴起的学科，其主要研究内容就是让计算机具有人类的分类能力。图像特征识别是指针对图像的模式识别，让计算机区分图像中的人脸、汽车、行人等目标就可看作图像特征识别。

进行图像特征识别时，首先要从图像中提取出代表该图像的数据特征，本文第四章中介绍的 LBP 特征描述就是一种具体的提取图像数据特征的方法。图像数据特征一般为多维向量，图像特征识别就是按照某种分类规则或者函数将不同的特征向量划分到预先定义好的不同类别中的过程。为了达到较准确的分类，进行图像特征识别之前一般需要通过训练得到这一分类规则或者函数。

目前用于图像特征识别的算法很多，但是往往运算复杂，在传统视觉芯片上难以实现。下面我们将分析 SVM、SOM, AdaBoost 这三种在图像特征识别中较为常用的算法，并分析它们在 HERO 上并行实现的可能性和方法。

5.2 支持向量机

支持向量机是 90 年代中期发展起来的基于统计学习理论的一种机器学习方法，通过寻求结构化风险最小来提高学习机泛化能力，实现经验风险和置信范围的最小化，从而达到在统计样本较少的情况下，亦能达到良好统计规律的目的。

图 5.1 为在二维平面上对两类点进行分类的示意图。其中红色的分割直线将红蓝两类数据点均匀的分割开来。该分割直线由几个关键点（黑色边缘点）唯一确定，且其到这些点的距离都为 h 。这些关键点被称为“支持向量（Support Vector）”，这些支持向量对分类直线起到了“支撑”作用。支持

向量机的求解、证明过程就是通过样本集来寻找分割正负样本的最优平面

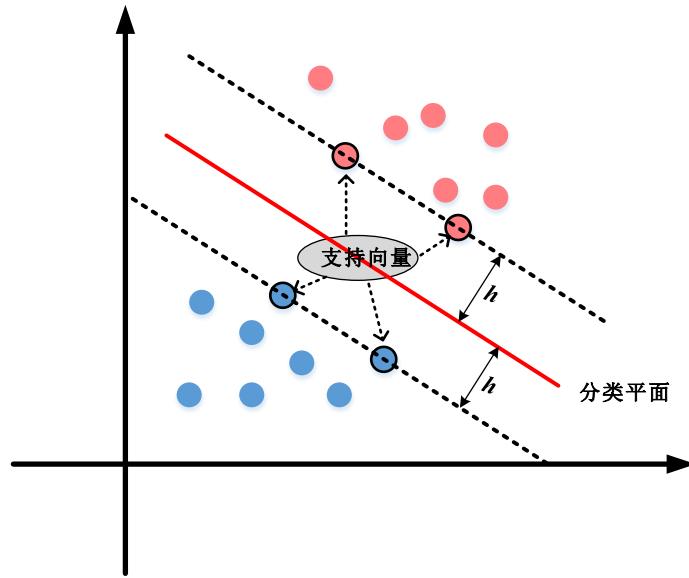


图 5.1 支持向量机中支持向量示意图

的过程，而该最优平面则由数据集中的支持向量确定。该过程所涉及的数学原理较为复杂，本文不再详细介绍，而是直接给出该最优平面，也就是分类函数的最终表达：

$$f(z) = \text{sign}\left(\sum_{i=1}^{\text{totalSV}} y_i a_i \kappa(x_i, z) - \rho\right) \quad (5-1)$$

上式中，支持向量（Support Vector, SV）是指对该最优平面的计算做出贡献的样本所对应的向量。 y_i 为支持向量 x_i 所对应的标签（Label），其值为 +1 或者 -1。 a_i 为支持向量对应的系数。 $\kappa(x_i, z)$ 为核函数， ρ 为一个阈值。一般情况下，我们可以认为核函数为线性函数，也就是向量内积：

$$\kappa(x, z) = \langle x, z \rangle \quad (5-2)$$

那么式 (5-1) 可以改写为：

$$f(z) = \text{sign}\left(\sum_{i=1}^{\text{totalSV}} y_i a_i \langle x_i, z \rangle - \rho\right) \quad (5-3)$$

对于任意待分类特征 z ，只需要计算它与支持向量的内积，并乘上 y_i 和 a_i ，就可以对其进行分类了。式 (5-3) 是可以进行分布式并行计算的，但是由于 Lane ALU 中没有乘法器，进行乘法计算时需要使用多次串行加法

来代替，将导致性能降低，为此本文没有采用支持向量机作为分类器。

5.3 面向视觉芯片的 SOM 神经网络模型

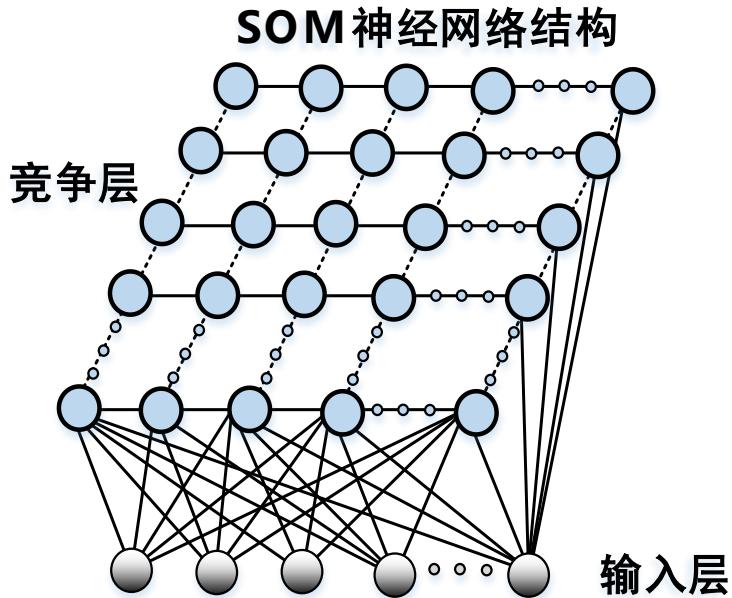


图 5.2 SOM 神经网络

SOM 神经网络是一种于上个世纪 90 年代提出^[66]的一种人工神经网络。通过训练，它能够得到输入训练集的二维离散表达。SOM 神经网络由 SOM 神经元组成，每个神经元中存储有与输入特征向量维度相同的参考向量，根据输入向量模式的不同，神经网络在不同的区域会产生不同的响应，其中一些区域的神经元被激活产生较强的相应，而其他区域中的神经元则受到抑制。训练过程中，SOM 神经网络不断的对激活和抑制的神经元及其周围神经元进行修改，训练完的神经网络能够对新的输入模式进行分类。SOM 神经网络的结构如图 5.2 所示，它包含输入层和竞争层（同时也是输出层），其中输入层的神经元数目为 N ，竞争层是由 $M \times M$ 个 SOM 神经元组成的二维网络，任意输入层神经元和任意输出层神经元之间互连。进行特征分类时，一个 K 维度的待分类向量将被广播到所有 SOM 神经元中，当某个神经元中的参考向量和输入向量相似时，该神经元就会被激活。反之，若输入向量和神经元中参考向量不相似，则神经元被抑制。根据激活神经元在二维网络中的位置，我们可以确定输入向量的类别。

为了进行 SOM 神经网络训练，我们首先定义一个有 $N_1 \times N_2$ 个 SOM 神经元的 SOM 神经网络，对于该二维阵列中第 i 行 j 列的神经元 (i, j) 其参考

向量 \mathbf{RV}_{ij} 的维度为 K 。对该神经网络的训练由如下几个步骤组成：

- (1) 初始化 SOM 神经网络竞争层中所有神经元参考向量 \mathbf{RV}_{ij} , 对向量中每个元素赋予小的随机初始值, 并且像图 5.3 那样将神经网络划分为若干个非重叠的区域, 每个区域对应一个类别。

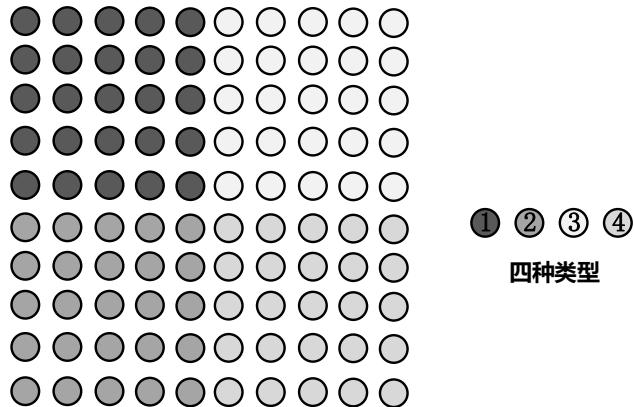


图 5.3 具有四种分类类型的 SOM 神经网络

- (2) 开始第 t 次训练, 从训练集中随机选择一个的特征向量 \mathbf{FV} , 并将其广播至 SOM 神经网络中;
- (3) 计算该输入向量 \mathbf{FV} 和所有神经元参考向量之间的的响应 VD_{ij} :

$$VD_{ij} = \| \mathbf{FV} - \mathbf{RV}_{ij} \| \quad (5-4)$$

- (4) 选择和 \mathbf{FV} 距离最小的神经元(i^*, j^*), 该神经元为获胜神经元:

$$\| \mathbf{FV} - \mathbf{RV}_{i^*j^*} \| = \min_{(i,j)} \{ VD_{ij} \} \quad (5-5)$$

- (5) 更新获胜神经元(i^*, j^*)及其邻域: 以神经元(i^*, j^*)为中心, 选择获胜邻域范围 $N(t)$, 邻域范围随着训练次数 t 的增加而减少, 直至只包含神经元本身。图 5.4 为一种可行的邻域选择方法: 在每完成 T 次训练后缩小一次训练范围。
- (6) 更新获胜领域的参考向量: 对于所有获胜领域 $N(t)$ 之内的 SOM 神经元, 对参考向量进行如下调整:

$$\mathbf{RV}_{ij}(t+1) = \mathbf{RV}_{ij}(t) \pm \eta(ND, t)(\mathbf{FV} - \mathbf{RV}_{ij}(t)) \quad (5-6)$$

其中 $\eta(t)$ ($0 < \eta(t) < 1$)为学习速率, 是一个随着训练次数和邻域距离增加而减小的函数, 一般将 $\eta(ND, t)$ 表示为两个独立函数 $\eta_1(ND)$ 和 $\eta_2(t)$ 之积。墨西哥草帽函数、大礼帽函数以及厨师帽等函数可以作为 $\eta_1(ND)$ 使用, 而指数下降、线性下降和分段线性下降函数可以作为

$\eta_2(t)$ 使用。当获胜神经元所在类别与输入特征向量的类别相同时，上式中 $\eta(ND,t)$ 前的符号取“+”号，反之取“-”号。

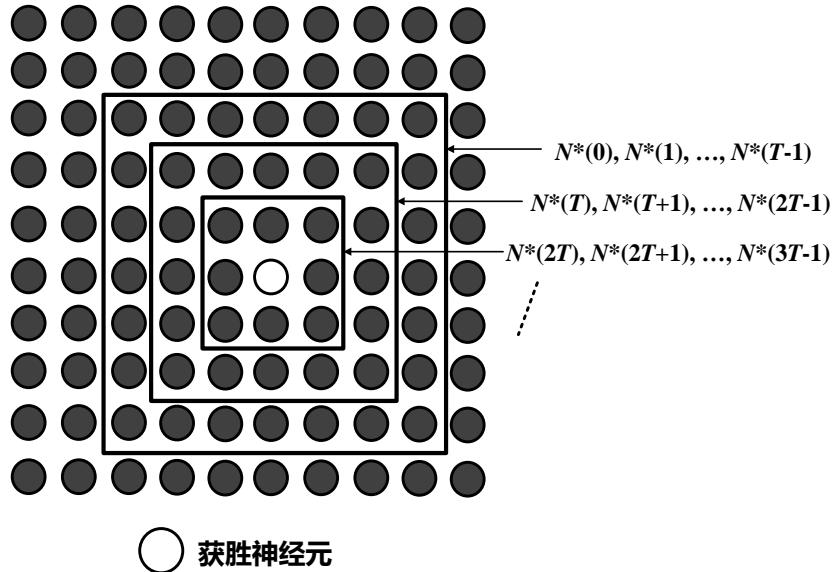


图 5.4 SOM 神经网络获胜邻域

- (7) 输入新的向量至神经网络中，并重复步骤（2）~（6），直到所有向量被输入了神经网络或者 $N(t)$ 或者 $\eta(ND,t)$ 减小至固定值；
经过上述步骤的训练可以得到一个具有分类能力的 SOM 神经网络，当新的特征向量输入该网络后利用第（3）和（4）步骤得到获胜神经元的坐标，该坐标所在的类别就代表了输入向量的类别。

要在硬件电路上实现 SOM 神经网络的训练和识别，我们利用 HERO 架构中每个具有多比特运算能力的 Lane 作为一个 SOM 神经元来完成式（5-4）~（5-6）的运算，并且进行邻域的选择和学习速率的调整。下面我们使用 HERO 硬件电路来对上述（1）~（7）步骤进行实现：

- (1) 初始化神经元参考向量。HERO 硬件无法产生随机数，参考向量由软件事先生成并存储器在局部存储器中。HERO 架构中没有设计浮点运算单元，但是可以使用定点数的形式表示特征向量和参考向量。通过软件模拟证明，在类别较少的情况下使用 16 位定点数表示参考向量和特征向量即可满足算法对数据精度的要求。
- (2) 广播特征向量。特征向量的广播可以使用 PPU 的广播指令完成，利用该指令，可以将数据广播至所有 LANE 中。
- (3) 响应计算。式（5-1）中使用 2 泛数表示特征向量与参考向量之间的距离，2 范数计算可表示为：

$$VD_{ij} = \sqrt{\sum_{p=0}^{K-1} (FV(p) - RV_{ij}(p))^2} \quad (5-7)$$

具体实现时,为了减少计算量和计算复杂度,可以使用街区距离近似:

$$VD_{ij} = \sum_{p=0}^{K-1} |FV(p) - RV_{ij}(p)| \quad (5-8)$$

该实现方法仅需要加法、减法和绝对值运算,消除了平方和开方运算,可以利用 Lane 中的算术运算指令完成。使用街区距离对算法并不会产生较大的影响。

- (4) 确定获胜邻域。每个 Lane 作为 SOM 神经元计算出特征向量与参考向量之间的距离后,PPU 可以从计算结果中选取最小的值,MPU 再进一步处理得到获胜神经元的坐标。
- (5) 更新获胜神经元及其邻域。根据获得的获胜神经元位置,MPU 通过 PPU 设置相应的 Lane 以及其周边 Lane 中的标志位为“总是操作”而其他 Lane 中的标志位为“总是不操作”。
- (6) 更新参考向量。根据 MPU 所确定的获胜神经元位置相应的执行式 (5-6)。为了便于在 Lane 中实现 $\eta_1(ND)$ 和 $\eta_2(t)$ 函数,可以选择图 5.5 所示的厨师帽函数来实现 $\eta_1(ND)$,即较近的神经元取值为 1,而较远的神经元取值为 0。 $\eta_2(t)$ 函数的一种典型的实现方式是每完成 T 次训练后, $\eta_2(t)$ 减小一次。因为其值小于 1,我们可以使用移位累加的方法来逼近。比如, $\eta_2(t)=0.75$ 时,对于数据 D,有 $D \times 0.85 = D \times (1/2+1/4)=(D>>1)+(D>>2)$, 其中>>符号表示 Lane 中右移运算。在参考向量的更新过程当中,如果数据溢出则会出现计算错误。为此,需要在每次更新参考向量值后计算参考向量的是否发生溢出,如果发生溢出则需要进行截断操作,将数值设置为数据格式所能表示的最大值。这个过程在 Lane 中可以通过大小比较和条件操作来完成。

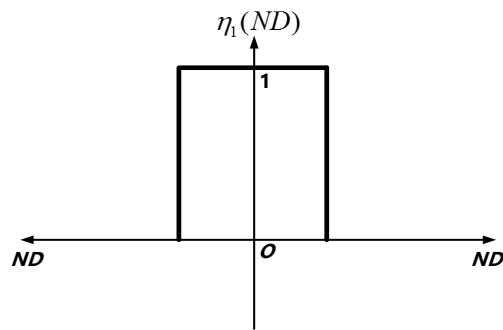


图 5.5 厨师帽函数

(7) 重复训练过程。重复训练过程就是重复执行训练的代码，由 MPU 和 PPU 控制程序指针跳转完成。

经过上述 (1) ~ (7) 步骤的分析，我们可以利用一个 Lane 完成 SOM 神经元的功能，来实现 SOM 神经网络的并行计算。相比于对于一个的串行实现 SOM 神经网络，其性能的得到极大的提升。

5.4 面向视觉芯片的自适应提升算法

自适应提升算法 AdaBoost，是英文"Adaptive Boosting"（自适应增强）的缩写，是由 Yoav Freund 和 Robert Schapire 于 1995 年提出的一种机器学习算法，两位作者也因此获得了 2003 年的哥德尔奖（Gödel Prize）。AdaBoost 算法可以与多种学习算法联合使用以提升它们的性能。它可以将多个其他学习算法（弱分类器）的输出通过加权求和的方式组合起来以得到一个经过提升（Boosted）的分类结果。它的自适应在于：前一个基本分类器分错的样本的权重会得到加强，加权后的全体样本再次被用来训练下一个基本分类器。同时，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数。AdaBoost 算法几乎不存在其他分类器常遇到的过拟合（Overfitting）问题，并且它对每个弱分类器的要求很低，只要稍微超过随机猜测准确率即可。

整个 Adaboost 的迭代算法可以用初始化训练样本权值分布、训练弱分类器、将训练得弱分类器组合成强分类器 3 个步概括。AdaBoost 算法对每个样本设置一个权值，初始时，每一类样本中每个样本的权值相同。在训练时，如果某个样本已经被前面的分类器准确地分类，那么在构造下一个训练集时，它的权值就被降低；相反，如果某个样本没有被准确地分类，那么它的权值就得到提高。这样的权值更新过程保证了没有被正确分类的样本在下一次分类时更加受“重视”。权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。每一轮训练都会得到一个弱分类器和其相应的权重因子。训练结束后，将这些弱分类器和其权重因子组合得到最后经过提升的强分类器。

对于用户已经取得的包含足够数量的已标记样本 $(x_1, y_1), (x_2, y_2) \dots, (x_n, y_n)$ ，其中实例 $x \in \chi$ ，实例空间 $\chi \subset \mathbb{R}^n$ ，正负样本分别用 $y_i=1$ 和 0 表示。AdaBoost 算法的具体流程如下：

(1) 对每一个样本 x_i 分别分配权值：

$$w_{1,i} = \begin{cases} \frac{1}{2m}, & \text{if } i=0 \\ \frac{1}{2n}, & \text{if } i=1 \end{cases} \quad (5-9)$$

其中 m 为样本中负样本的数目, n 为样本中正样本的数目。对正负样本权值的初始化方法使得所有样本的权值之和为 1, 并且正样本和负样本的权值之和分别为 $1/2$ 。

(2) 对于 $t = 1, \dots, T$.

1. 归一化权重因子

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \quad (5-10)$$

使 w_t 中 $w_{t,i}$ 之和为 1, 使其表现为概率分布的形式。

2. 对于每个特征, j , 训练一个弱分类器 h_j 并限定该分类器只使用该特征, $h_j(x)$ 的输出为 0 或者 1。对于本次样本集的权值 w_t , 可以得到误差函数:

$$\varepsilon_t = \sum_i w_i |h_j(x) - y_i| \quad (5-11)$$

3. 选择弱分类器, h_t , 使其的误差函数值 ε_t 最小

4. 更新权值:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i} \quad (5-12)$$

其中, 如果样本 x_i 被正确分类, 那么 $e_i = 0$, 否则 $e_i = 1$, $\beta_t = \frac{\varepsilon_t}{1-\varepsilon_t}$ 。

(3) 最后训练得到的强分类器为:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^T a_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T a_t \\ 0 & \text{otherwise} \end{cases} \quad (5-13)$$

其中, $a_t = \log \frac{1}{\beta_t}$

上述 AdaBoost 训练算法颇为复杂, 为了保证最后强分类器的正确性, 必须采用浮点数的形式进行运算, 并且计算过程中涉及 \log 、幂数、除法等复杂运算。为此, 我们可以采用线下训练的方式得到强分类器。事实上,

AdaBoost 训练所需要的样本量和计算量都非常大，我们发现使用配置为 Intel i5 处理器，12G 内存的计算机训练也需要数十分钟才能完成一次训练。如此规模的计算量在硬件资源极其有限的视觉芯片上实现是不现实的。因此，对于 AdaBoost 算法使用线下训练的方式，在视觉芯片上仅完成识别。进行特征识别时，将特征向量 x 带入式（5-13）根据计算结果即可完成分类。为了方便表示，（5-13）可以用如下形式表示：

$$C(x) = \text{sign}\left(\sum_{t=1}^T a_t h_t(x) - T\right) \quad (5-14)$$

其中，

$$T = \frac{1}{2} \sum_{t=1}^T a_t \quad (5-15)$$

经过线下训练，弱分类器以及弱分类器的权重因子已经确定，所以 T 可以看做一个常数。对于弱分类器 $h_t(x)$ ，可以使用一个阈值分割函数来完成以尽量减少计算量和计算复杂度：

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j < p_j \theta_j \\ 0 & \text{otherwise} \end{cases} \quad (5-16)$$

上式中， f_j 表示特征， θ_j 表示阈值， p_j 代表不等号的方向，其值为 +1 或者 -1。阈值 θ_j 为将最小样本分错的一个最优值。式（5-16）意味着，弱分类器 $h_j(x)$ 可以由一个阈值 θ_j 和 p_j 表示，并且计算弱分类器结果时，只需要比较大小和取法操作即可完成。软件模拟结果表明，使用 16 位定点数来表示式（5-14）中的数值完全可以满足精度要求，对分类结果的正确性不会造成影响。

使用 AdaBoost 提升得到的强分类器进行特征分类时，首先将所有用到弱分类器参数都存储到局部存储器中，根据弱分类器的多少每个 Lane 可以进行一个或者多个弱分类器计算。进行特征分类时，所有 Lane 可以同时计算 $a_t h_t(x)$ ，并在数十个时钟周期内得到计算结果。PPU 对其下所有 Lane 的计算结果求和，最后 MPU 得到强分类器分类结果。

5.5 本章小结

本章简要介绍了图像特征识别的常用方法，分析了 SVM 在 HERO 架构中应用的可能性，然后重点研究了 SOM 神经网络的训练和识别算法模型，并分析了如何在 HERO 硬件电路上并行实现 SOM 训练和识别的关键

步骤。本章还对 Adaboost 的算法流程进行了详细的介绍并阐述了其并行实现方法。

第6章 多级异构并行处理器的实现与测试

6.1 多级异构并行处理器的实现及测试系统

本文使用 Arria V^[67] FPGA 实现了 HERO 架构。本文侧重于视觉芯片处理器架构设计，因此没有独立设计视觉芯片高速图像传感器，测试系统中我们使用了一款商用高速摄像头。HERO 架构的 FPGA 实现中，包含了 4 个计算核心，每个计算核心由一个主处理器和三个虚拟处理器组成。每个主处理器和虚拟处理器下都有一个由 32 个 Lane 组成的 Lane 阵列。每个 Lane 同时又等效于 16 个 PE，因此一个 Lane 阵列构成了 16×32 的 PE 阵列。为了便于后续的 ASIC 实现，除存储器模块外，HERO 架构的其他所有模块的实现均使用了可综合的 Verilog HDL 语言描述。表 6.1 为 HERO 实现的一些具体参数和性能。HERO 测试系统主要由相应的 FPGA 开发板、商用摄像头、PCB 子板和外部测试软件构成。

表 6.1 HERO 架构 FPGA 现实以及性能参数

FPGA	Arria V
逻辑资源使用率 (in ALM)	123,447/136,880 (90%)
寄存器资源使用率	67023
存储器资源使用率	4040776/17674240 (23%)
系统时钟频率	50MHz
PE 阵列大小	$16 \times 32 \times 16$
Lane 阵列大小	$32 \times 1 \times 16$
PPU	16 位 MIPS $\times 16$
局部存储器	4KB $\times 16$
控制处理器	32 位 MPU
峰值运算性能	64GOPS@8 比特数据

为了方便实现第四、五章所述各种算法以及传统的图像处理功能，我们基于 Visual C# 语言开发了集成开发环境（Integrated Development Environment, IDE）。IDE 的主要功能为汇编（其中汇编器使用 Perl 语言开发）并且下载在 HERO 上执行的程序代码，各种配置参数、测试图像和视频序列等，最后读回并显示视觉芯片的处理结果。HERO 系列视觉芯片的算法程序开发过程与传统的 C、C++ 程序有所不同，它属于 C 和汇编语

言混合编程，MPU 上的编程属于 C 语言编程，而 PE、Lane 以及 PPU 则使用第三章描述的指令集进行汇编语言编程。如图 6.1 所示，整个系统的工作是由 MPU 控制完成的，它通过调用底层的驱动函数使 PE、Lane 以及 PPU 的汇编程序在相应的硬件上执行。MPU 程序和汇编程序分别编译，最后通过下载器下载到视觉芯片上。

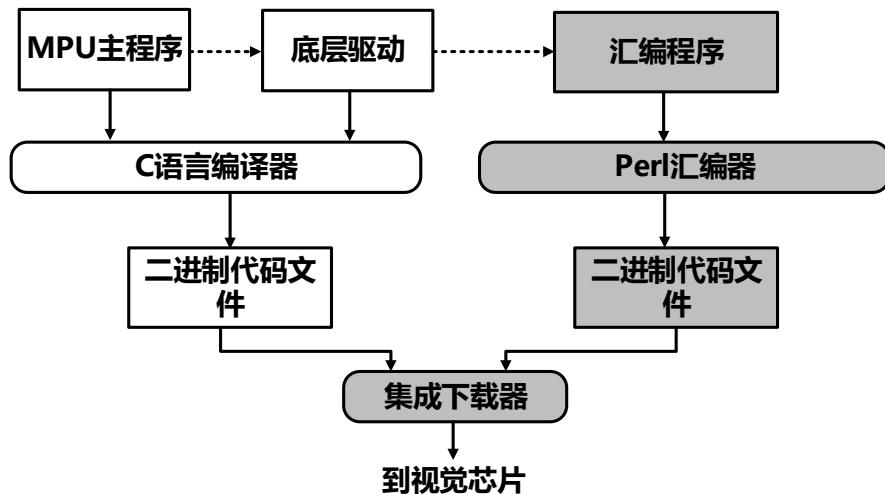


图 6.1 视觉芯片代码编译流程

6.2 特征点检测与特征描述算法测试

FAST 与 SIFT 角点检测的测试结果如图 6.2 所示，图中“.”标识的像素点为检测到的特征点^[49]。从图中我们可以看出，FAST-9 特征点的检测中，实验所得到结果和使用作者原始代码^[55]所获得结果一致。该结果证明了本文第五章中所提出的 FAST 角点在 PE 阵列上的并行实现方式是有效可行的。经过 PE 阵列并行化后的 SIFT 角点算法由于缺乏浮点精度，并且使用近似的高斯函数，所以计算结果和使用作者代码^[5]所得检测结果有一定的差异。尽管如此，本实验仍然检测出了图中大部分具有代表性的特征点，比如眼轮廓、鼻尖、帽沿以及发梢的特征点。图 6.3 中为 PE 阵列执行第五章所述 SIFT 与 FAST 特征点算法所使用的 PE 操作数。从图中我们可以看出完成 FAST 角点所需的指令周期远远小于 SIFT，其原因在于 SIFT 需要进行多次高斯滤波，为此消耗大量指令周期。这两种算法都大量使用比特加法和逻辑与运算，SIFT 算法由于使用了第五章中将二维滤波拆分水平和垂直两个方向一维滤波的方式，使得数据转移（Data-Transfer）指令相对较少，而 FAST 指令由于需要与较远距离的像素进行大小比较，所以数据转移指令相对较多。



图 6.2 FAST 与 SIFT 角点检测算子测试结果

表 6.2 将本文系统的性能和最近几年较有代表性的特征点检测硬件^[60, 68, 69]进行了对比。对于 512×512 分辨率的图像，使用第四章中所述并行实现完成 SIFT 和 FAST 算法分别能够达到 460fps 和 30fps 的系统性能。相比于 Moko 等人使用嵌入式处理器的实现方式，FAST 特征点检测的性能提高了 16 倍。对比 Clemons 等人使用多核处理器的 FAST 与 SIFT 实现方法，虽然其处理图像分辨率更大并且 SIFT 特征点处理性能也更高，但

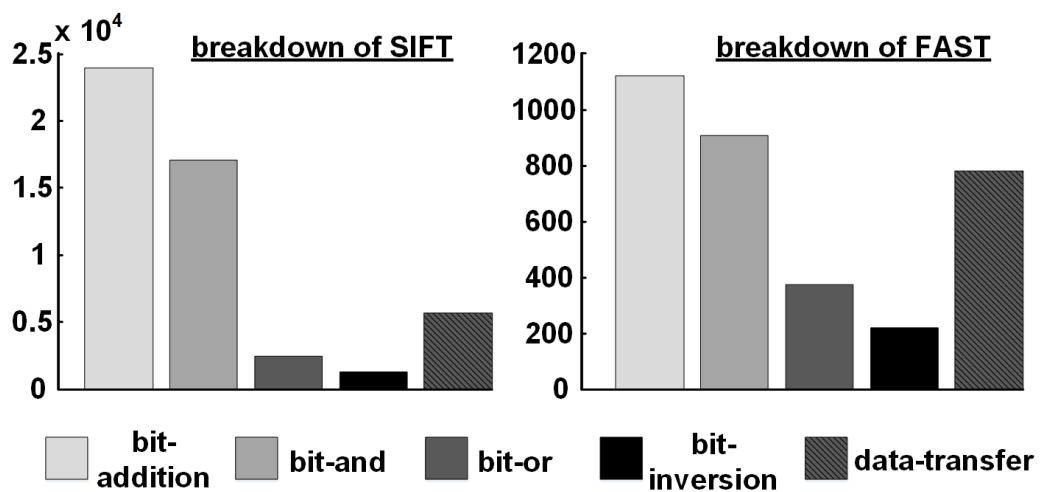


图 6.3 SIFT 以及 FAST 特征点检测指令组成

是其系统频率高达 1GHz，如果要达到与之相同的性能，HERO 只需要使用 200MHz 左右的主频即可，由此可以看出在并行性上我们的架构是有很大优势的。Dohi 同样使用了 FPGA 实现，对 640×480 大小的图像实现了每秒 62.5 帧的 FAST 特征点检测，其系统性能与我们的架构有明显的差距。使用 PE 阵列完成一次 SUSAN 角点检测算法需要约 7500 个周期，在 50MHz 的时钟下，完成 512×512 大小图像需要 10 ms，可以达到 100 帧左右的处理速度。

表 6.2 角点检测性能对比

	本文	Moko <i>et al.</i> [68]	Clemons <i>et al.</i> [60]	Dohi <i>et al.</i> [69]
Algorithm	FAST SIFT	FAST	FAST SIFT	FAST
Implementation	FPGA	Embedded processor	Multicore processor	FPGA
Frequency	50MHz	168MHz	1GHz	25MHz
Resolution	512×512	512×512	1024×768	640×480
Processing speed (fps)	FAST: 460 SIFT:30	27.3	Approx.40	62.5

本文第四章提出了 LBP 以及 HoG 两种特征描述子在视觉芯片上的实现方法。进行 LBP 特征描述时，使用 PE 阵列像素级并行的方法生成 LBP Pattern 需要约 800 个时钟周期。当然我们也可以采用 Lane 来进行 LBP Pattern 生成，对于一个 32×32 的图像块，需要 1500 个时钟周期，其效率相对于使用像素级并行降低了 1 倍。如图 6.4 所示，LBP Pattern 生成完成后，图像中每个像素被重新赋予 0~9 之间的值。直方图统计时，借助 HERO 架构中 PE-PPU 关系以及共享存储器机制，PPU 可以直接访问 PE 阵列生成的 LBP Pattern 数据，并进行直方图统计。若采用 3.3.2 节中的方式，仅仅需要 2000 余个周期即可完成，相当于每个像素的平均统计时间为两个时钟周期。HERO 进行 HoG 特征描述时，进行水平和垂直方向梯度计算时需要使用 40 个时钟周期，而求解 $|dx| + |dy|$ 时需要 56 个时钟周期。梯度方向性确定以及直方图统计则消耗较多时间，为 5500 个时钟周期。

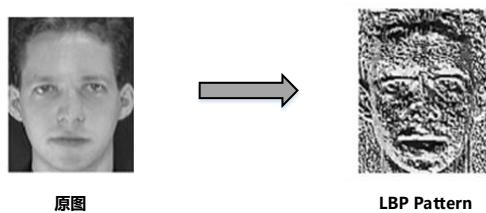


图 6.4 经过 LBP Pattern 生成后的图像

6.3 完整应用算法测试

为了验证 HERO 架构在进行局部特征表达和特征分类方面的整体性能，本文实现了人脸检测、人脸识别、其他目标识别以及目标追踪等完整应用算法。下面我们将对这几种算法以及测试结果进行介绍。

6.3.1 人脸检测

人脸检测的核心在于利用有效的特征向量进行人脸表达，并且使用可靠的分类器对特征向量进行分类。在本小节中，我们将结合局部图像特征和 AdaBoost 分类器来实现人脸检测算法。

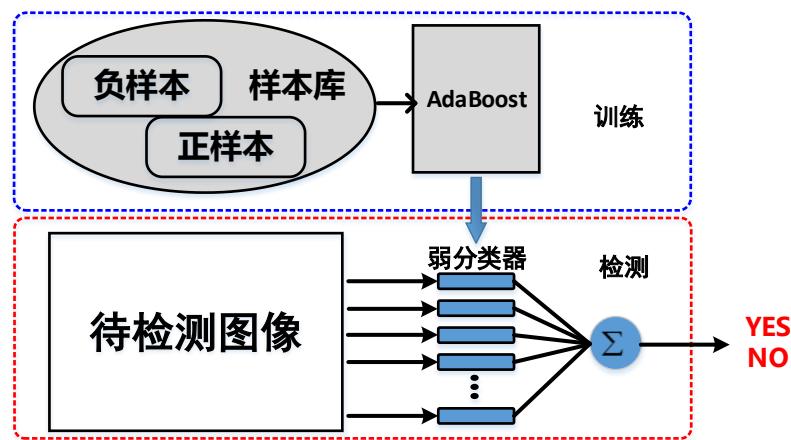


图 6.5 人脸检测框架

如图 6.5 所示，为本文进行人脸检测实验的算法框架。首先，我们利用包含大量正样本和负样本的样本库进行训练并从中选择出 N 个最能区分正负样本的弱分类器组合，其中每个弱分类器由一种图像特征以及相应的阈值组成。上述训练过程所选择出的 N 个弱分类器构成强分类器，当需要进行检测任务时，首先计算待检测图像中的 N 个图像特征并与相应的阈值比较，最后通过加权求和得到最后的识别结果。该算法框架中并没有指明使用哪一种局部图像特征作为弱分类器，事实上，根据应用的不同，该框架可以使用多种不同的图像特征。比如本章前面所提到的 LBP 和 HOG 特征，甚至图像中某个位置像素的灰度值都可以作为构成弱分类器的备选特征。

使用上述检测框架，待检测图像或者图像窗口大小应该和训练样本中图像大小一致，因此对于较大分辨率图像，在实验中我们可以采取“滑

动窗口（Sliding Window）”的方式来进行检测。如图 6.6 所示，首先对原始图像取固定窗口大小并通过特征提取和特征分类器判断该区域是否存在人脸，不断地在图像上滑动该窗口位置并重复上述过程，直至遍历整个图像。对图像进行亚采样，并重复上述过程直至亚采样后图像大小与窗口大小一致。使用“滑动窗口”方法可以保证检查到所有尺度的人脸图像。

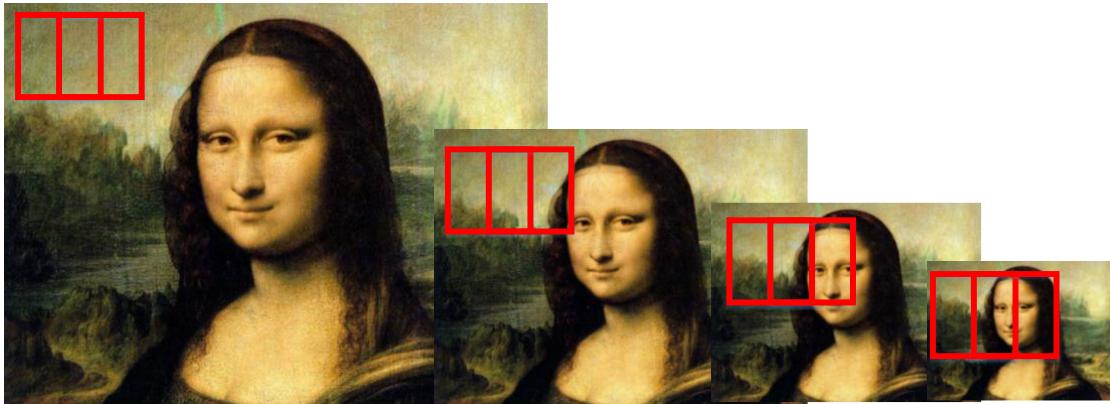


图 6.6 使用滑动窗口方法进行人脸检测

我们将 ORL 人脸特征数据库^[70]中 400 张 64×64 分辨率的图像作为正样本，并且随机选择了 400 张 64×64 大小的非人脸图像作为负样本用作样本库，图 6.7 中为部分正样本和负样本图像。所有样本均被划分为 8×8 个区域，每个区域的大小为 8×8 像素。我们对这些区域进行 LBP 特征表达，并对每个区域提取 LBP 直方图，其中每个区域的直方图维度为 10 维。最后我们将这 64 个区域的直方图合并为一个 640 维的特征向量。这 640 维的向量中任意一维都可看做一个弱分类器 $h_j(x)$ 。通过对所有样本的统计，对任意弱分类器 $h_j(x)$ ，我们可以选择一个最优的阈值 θ_j ，使其能够最好的区分正负样本。这样弱分类器可以用如下形式表达：

$$h_j(x) = \begin{cases} 1 & \text{if } f_j(x) \leq \theta_j \\ -1 & \text{otherwise} \end{cases} \quad (6-1)$$

上式中， $f_j(x)$ 为直方图第 j 个维度的值。通过使用 AdaBoost 算法，可以从这 640 个弱分类器中筛选出 100 个弱分类器组成强分类器，并且计算出其中每个弱分类器的权重。图 6.8，为这 100 个弱分类器的分布，其中有 31 个采用了 LBP 纹理中的 Pattern 4，为最多。这说明经过 AdaBoost 筛选出了弱分类器更多的利用边缘信息来区分人脸与非人脸。通过对这 100 个弱分

类进行加权求和，就可以得到分类结果。在进行人脸检测之前，我们将这 640 个弱分类器的阈值 θ_j 以及权重都存储在相应的 PPU 存储器中（非 AdaBoost 筛选出的弱分类阈值和权重皆为 0）。当 PE 阵列生成 LBP Pattern 图且 PPU 提取出局部直方图后，就可以直接使用式（2-7）、（2-8）所示的分布式并行的方法进行分类。最后所有 PPU 的结果被 MPU 累加，并与分类函数的阈值 T 比较，得到分类结果。



图 6.7 本文使用训练库中正负样本示意图

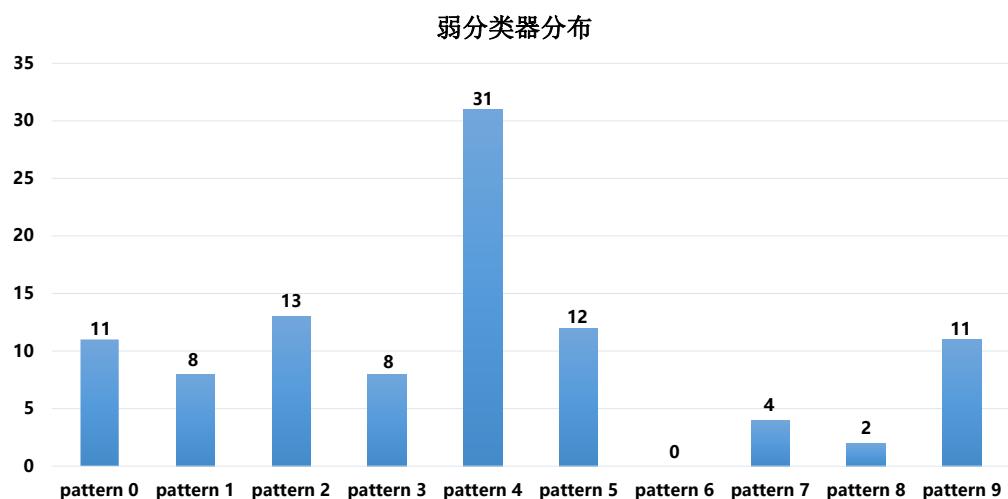


图 6.8 弱分类器在 LBP 各种模式中的分布

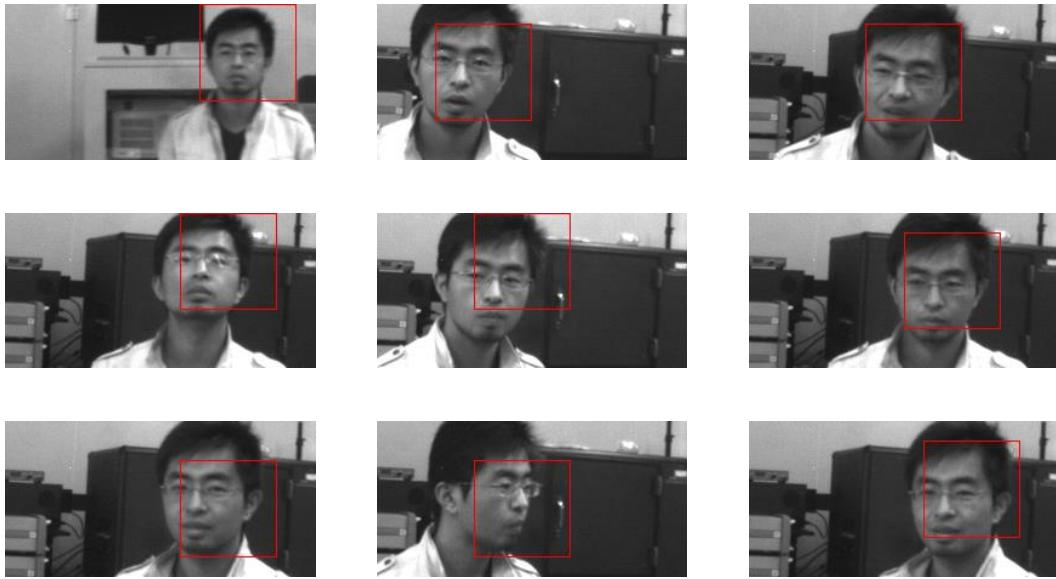


图 6.9 人脸检测结果

当系统运行在 50MHz 下时，完成一次 64×64 窗口大小的人脸检测算法只需要消耗 $60\mu s$ 。通过结合 HERO 架构中的 MIMD 工作模式和灵活的图像块选取方式，该系统可以高效实现对整个图像中的人脸搜索。在任意一次从图像传感器读出图像的过程中，可以将任意 256×64 大小的区域划分为 4 块 64×64 区域并存储在局域存储器中，对这 4 块图像处理耗时为 $4 \times 60 = 240\mu s$ 。在下一次读出时，对该 256×64 区域进行再次采样，并处理与前次处理交叠的 3 块 64×64 的图像。该过程耗时为 $3 \times 60 = 180\mu s$ 。使用这种方式，可以对图像传感器阵列中所有区域进行扫描。图 6.9 显示了部分检测结果。为了对人脸检测的识别正确率进行测试，我们使用了 LFW 人脸数据库^[71]进行测试，该数据库包含 200 幅各种场景下的人脸图像，我们将这些人脸图像作为图像传感器数据输入，并进行了人脸检测处理。系统能够达到 96% 的识别准确率，部分检测结果如图 6.10 所示。

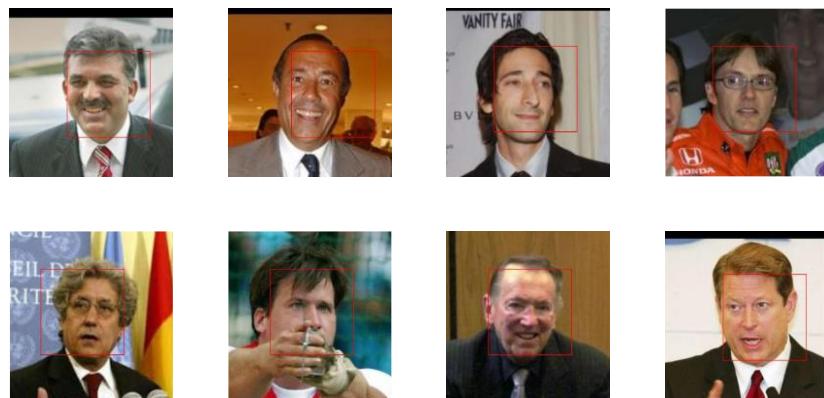


图 6.10 LWF 人脸库检测结果

本文还使用了 HoG 特征和 LBP，HoG 混合特征进行了人脸检测实验，同样获得了类似的实验结果。图 6.5 中人脸检测的架构可以在 HERO 中快速的实现，同时还具有很高的通用性和灵活性。通用性是指，我们可以通过简单的替换样本库来实现对不同物体的检测与识别并且可以达到和上述实验类似的处理速度。灵活性是指根据被检测物体的特点可以选择更加合理的局部特征以提高识别的准确率。

6.3.2 人脸识别

用于人脸识别的算法有特征脸算法、Fisherface 方法以及基于 LBP 描述的方法。根据文献[43]中报道的结果，基于 LBP 描述方法的人脸识别率最高，因此本文采用了 LBP 特征进行人脸描述，整个人脸识别的流程如图 6.11 所示，首先利用 LBP 局部特征描述方法提取出人脸特征，然后通过 6.3.1 节中得到的人脸检测分类器判断该特征是否为人脸。如果检测结果非人脸，那么终止算法，否则将得到的特征向量输入人脸库中进行寻找与其最为匹配的人脸。

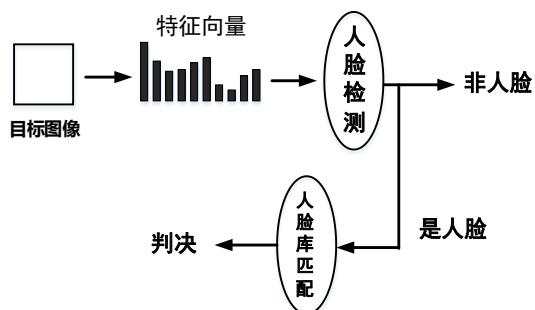


图 6.11 人脸识别流程

本文首先使用了 SOM 神经网络来完成人脸识别，我们仍然采用 ORL 人脸数据库进行测试，受限于 HERO 架构中神经网络的规模，无法实现大量人脸的分类，因此我们选取了 ORL 库中 7 个人脸构建人脸库，并训练了相应 SOM 神经网络，该网络的人脸识别正确率约为 87%。利用 SOM 神经网络进行人脸识别所消耗的时间在 1ms 以内，其分类结果如图 6.12 (b) 所示。

针对每一个人脸样本，我们还可以利用 AdaBoost 训练一个强分类器，用于区分该人脸和其他人脸。我们使用 ORL 人脸数据库中 40 个人脸，每个人脸 10 张图像训练了 40 个分类器，对所有人脸识别正确率为 85.8%。受限于 HERO 架构中存储器大小，进行实际测试时，可存储 10 个左右的分类器用于人脸识别，由于需要多次运行分类器，执行整个算法的时间会与上一小节中人脸检测算法呈线性关系，部分分类结果如图 6.12 (c) 所示。

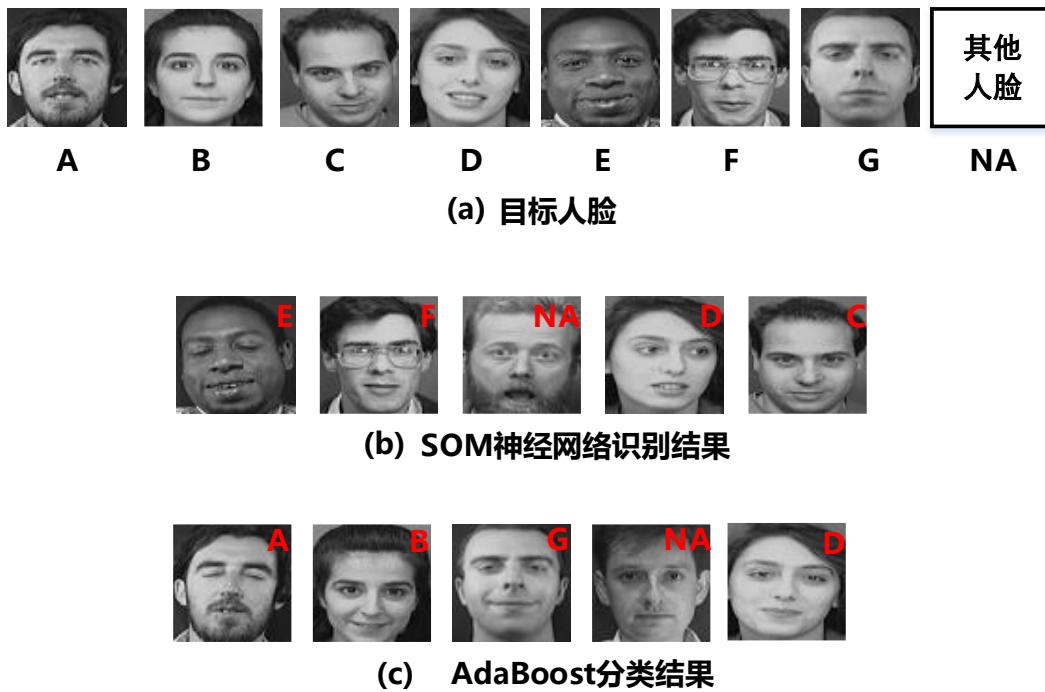


图 6.12 部分人脸识别结果

上述人脸识别测试旨在验证 HERO 架构的有效性，因此没有深入的研究人脸识别算法，识别能力与一些文献相比存在一定的差距。但是，通过使用更加具有区分性的人脸特征、提高神经网络的规模以及训练更强的分类器等方法，可以在 HERO 架构上实现更好的人脸识别效果。

6.3.3 其他目标识别

本文还进行了其他目标识别的测试。图 6.13 中吉普车、坦克、水泥路面以及草地为四类待识别目标，每类目标皆含 1 万个样本，样本图像大小为 256×256 。我们分别使用了 SOM 神经网络和 AdaBoost 两种分类方法对这四类目标进行了分类。

对 SOM 神经网络进行训练时，首先将 SOM 神经网络非交叠的分成了四个部分，然后经过有监督的训练而得可用于分类的 SOM 神经网络。我们使用了每类 1000 个测试样本，共 4000 个测试样本对训练所得 SOM 神经网络进行了测试，系统正确率约为 83%。HERO 架构中，完成上述识别过程耗时在 1ms 以内。

本文使用图 6.14 中的方式将多个 AdaBoost 训练的二分类器组合来实现多类别分类。在训练分类器 1 时，我们将坦克和吉普车的样本作为正样本，而水泥路

面和草地的样本作为负样本。训练分类器 2 时，将吉普车样本作为正样本，坦克作为负样本。将水泥路面和草地分别作为正、负样本得到分类器 3。进行特征分类时，特征向量首先经过分类器 1 分类，然后根据分类结果在决定使用 2 或者 3 分类器。得到训练完成的三个分类器后，仍使用每类 1000 个测试样本，共 4000 个测试样本对分类效果进行了测试，分类正确率达到了 90% 以上。

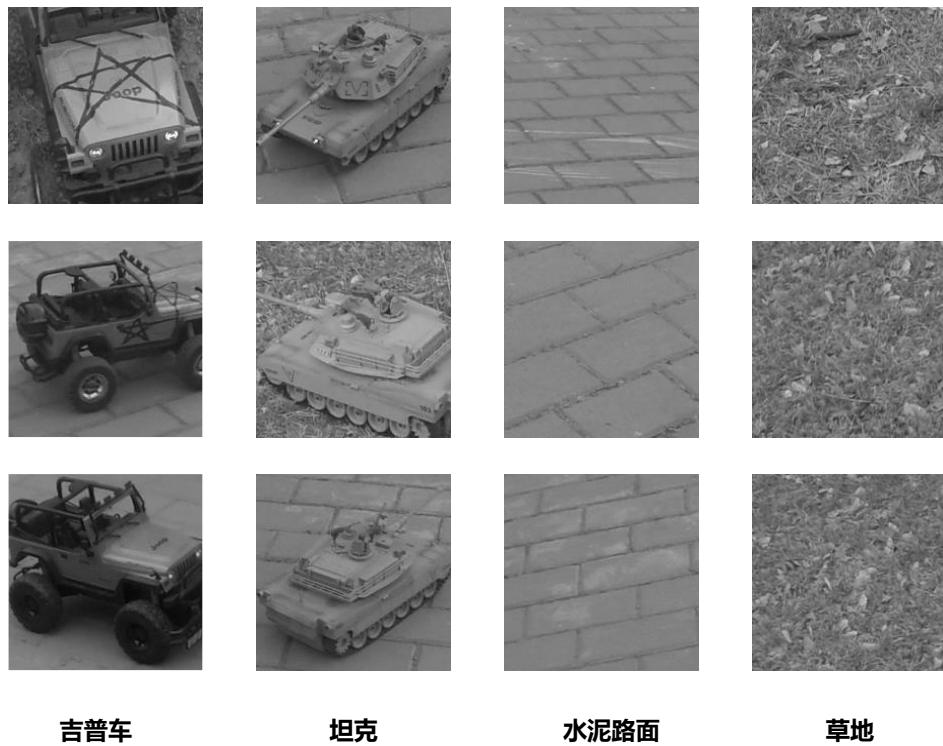


图 6.13 四类待分类目标

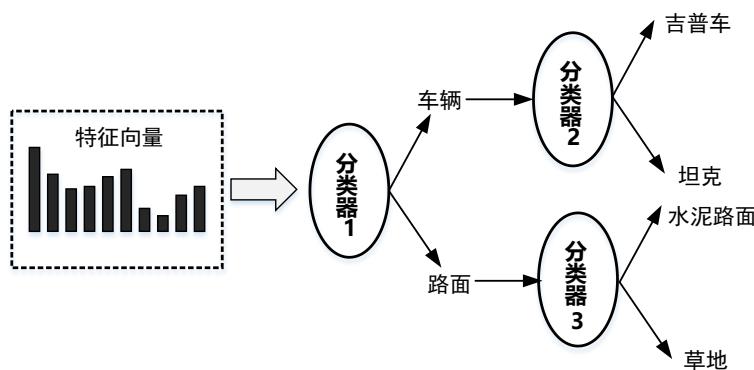


图 6.14 利用 AdaBoost 完成多分类

在 HERO 架构上利用 SOM 神经网络进行多目标识别时，其用于识别特征的时间和二分类时相当。但是，随着类别的增加，在不增加神经网络规模的情况下，其识别分类能力将会逐渐降低。使用 AdaBoost 进行多分类时，随着目标数目的增加，需要执行多个分类器，因此算法消耗的时间将会有一定增加，并且为了保存

多个分类器的参数，也会使用一定的存储器空间，但是其分类效果一般不会随着类别的增加而明显降低。在 HERO 架构上，我们可以灵活的选择或者组合这两种分类器以实现分类效果、性能、速度等多个方面的良好折衷。

6.3.4 目标追踪

利用视觉芯片来开展目标追踪的研究已经开展了数年，然而到目前为止视觉芯片仍然很难完成在复杂背景和光照变化情况下的目标追踪任务。目标追踪的效果取决于目标模型建立的好坏。已经报道的视觉芯片实现中，一般假设目标和背景的灰度值相差明显，这样仅仅通过灰度值就能很快区分目标和背景。然而，在实际的目标追踪，尤其是高速目标追踪中，目标和背景灰度值常常会发生剧烈的变化，在这种情况下如果仍然使用灰度值作为区分目标与背景的特征，目标将很容易在追踪过程中丢失。为实现具有鲁棒性的高速目标追踪任务，必须对目标使用更具有描述性的特征而非简单使用灰度值区别。

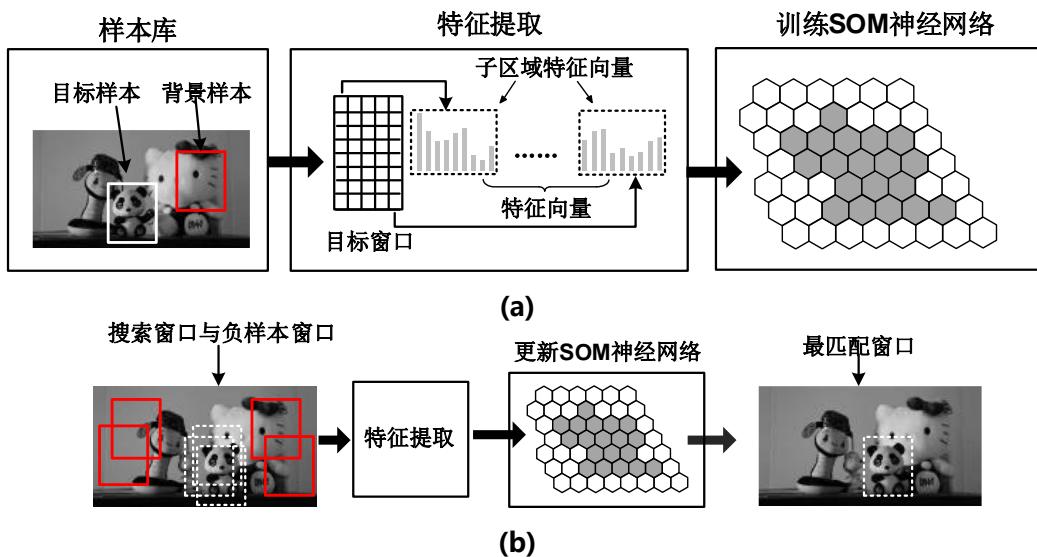


图 6.15 目标追踪算法

图 6.15 为本文所提出用于高速目标追踪的算法框架。图 6.15(a)所示，在进行目标追踪任务前，使用 4.2 节中任意特征描述方法对样本库中目标进行特征描述。使用局部特征描述方法比使用全局灰度特征更加具有鲁棒性，尤其 LBP 算子对光照变化不敏感，使其非常适合光照变换下目标的特征描述。利用提取的特征向量训练一个对背景和目标进行分类的 SOM 神

经网络。在完成了上述过程后，我们就可以开始对标定目标进行追踪了。如图 6.15 (b) 所示，系统开始工作后，我们首先在上一帧目标窗口的周围选择数个搜索窗口，在远离目标窗口的位置选择数个背景窗口，对这些搜索窗口和背景窗口进行特征提取，这样我们可以得与窗口数量一致的多个特征向量。将这些特征向量输入 SOM 神经网络，根据上一帧的神经网络判断每个特征是否为目标，并同时更新神经网络为下一帧的分类做准备。不断重复图 6.15 (b) 中的过程即实现了对目标的追踪。

上述目标追踪框架具有两个特点。第一，对于目标的描述采用了动态更新方式，即每一帧中都会根据目标位置采集一定数量的样本用于 SOM 神经网络的更新，动态更新保证目标在逐渐发生旋转、尺度变换的过程中神经网络模型也随之更新。第二，因为本算法完成一帧追踪的时间能够保持在 1ms 以内，因此每两帧之间目标在图像中位移能够保持在几个像素之内。因此，我们可以将搜索窗口的位置集中在上一帧目标位置附近，以减少搜索过程中的计算量。如图 6.16 所示，蓝色虚线框表示上一帧中计算出的目标位置，而红色虚线框表示新的一帧中对目标进行搜索的多个搜索框。进行追踪时，需要计算所有搜索框中的目标特征，如果对每个搜索框都进行独立编程计算，这一过程将变得异常复杂繁琐，并且会增加程序存储器的负担。为了简化这一过程，本文提出一种利用数据移动来等效搜索框移动的方法。如图 6.16 中，对于左下角的红色搜索框，我们利用对数据在水平以及垂直方向进行 dx_1 , dy_1 像素的移动，这相当于将搜索框中的数据移动到了目标窗口的位置，这时我们就可以重用对目标窗口进行特征提取的程序来完成搜索窗口的特征提取。

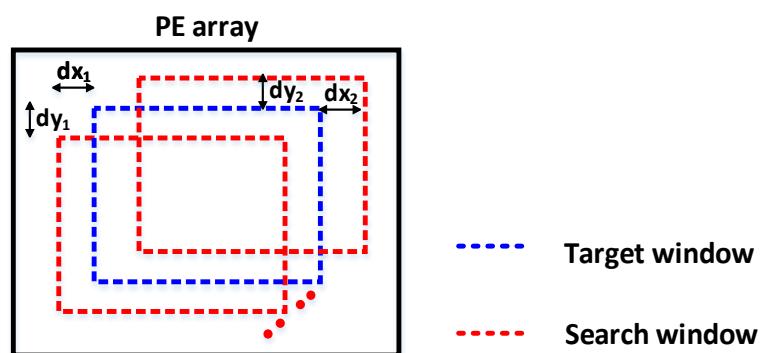


图 6.16 使用 PE 阵列快速实现搜索框确定

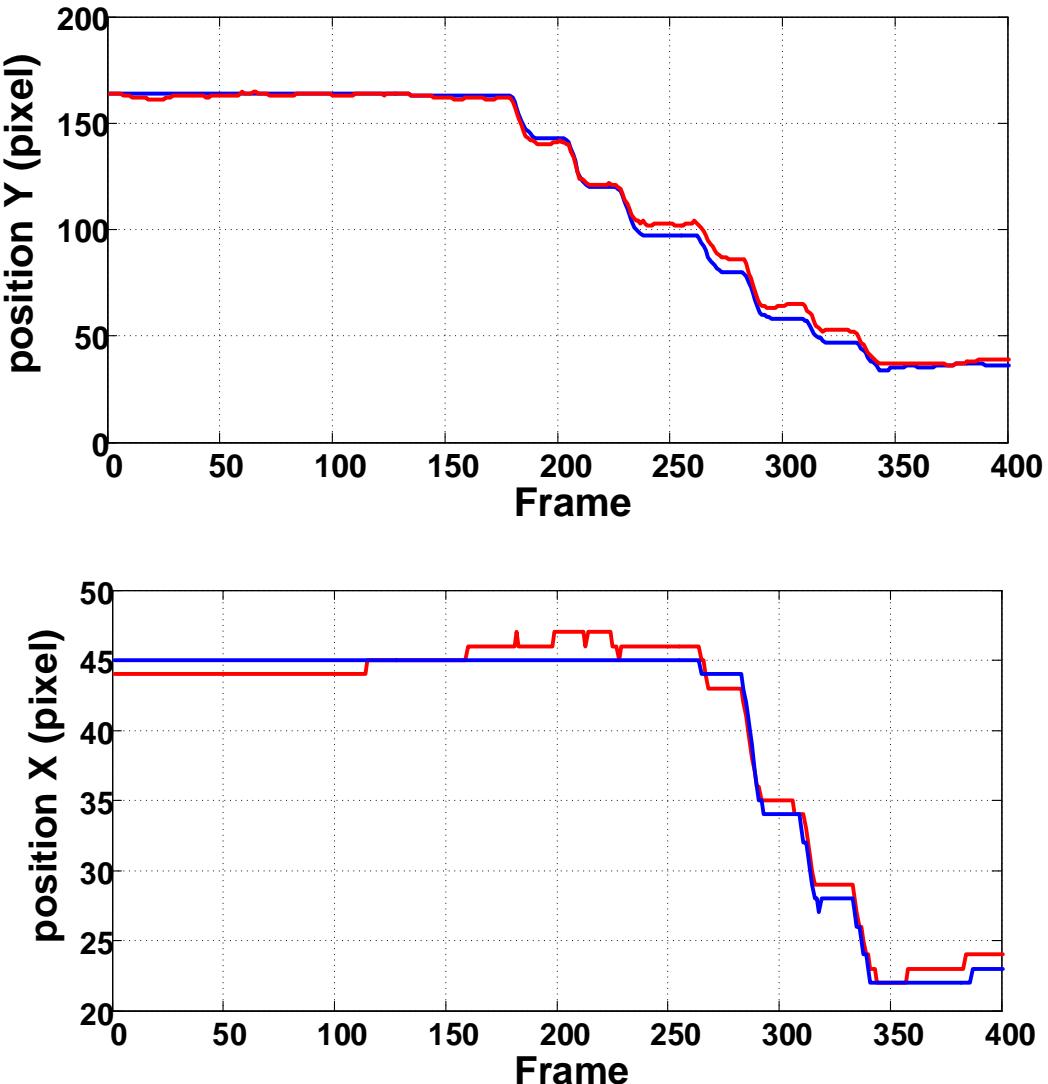


图 6.17 追踪算法仿真结果

我们使用了包含 400 张图像的测试序列对追踪算法进行了测试。图 6.17 为该序列的测试结果，其中蓝色和红色线表示目标实际位置和算法预测位置，在 400 帧的追踪中，算法在垂直（ y 坐标，上图）以及水平（ x 坐标，下图）方向都能够正确的寻找到目标位置。

图 6.18 所示，为进行目标追踪的实验装置，包括一个二维自由度转台，图像传感器，FPGA 以及实验用小车。实验开始前，首先借助 IDE 进行被追踪目标选定。实验开始后每一帧，HERO 架构根据图像特征搜寻最佳的匹配窗口并将其作为目标位置，MPU 计算该位置与图像传感器视场中心位置在水平以及垂直方向上的偏移量 dx 和 dy 。这两个偏移量随后被发送至马达控制器模块驱动马达在两个方向上运动以使目标重新回到视场中央。整个实验过程中，如果追踪算法鲁棒性强，那么目标将不会丢失，并且将

保持始终处于图像传感器中心位置附近，反之如果追踪算法鲁棒性较差，那么目标将很快逃出图像传感器视野范围。



图 6.18 目标追踪实验装置

图 6.19 为使用图 6.18 中测试平台在实验室环境下的测试结果。在光照闪烁、障碍物以及目标旋转、尺度变化、甚至摄像头成像质量较差的情况下，算法仍然能够对目标进行跟踪，并将其固定在图像传感器视场中央位置。

图 6.20 分析了在 HERO 架构上进行目标追踪所使用的指令周期，PE 和 Lane 阵列消耗了 $194 \mu\text{s}$ ，完成了 10.8M 运算，而 PPU 阵列在 $309\mu\text{s}$ 的时间内完成了 2.1M 运算，MPU 消耗了系统运行的大部分时间， $490\mu\text{s}$ ，但是由于其串行计算的本质，其完成的运算仅为 24.4K 左右。整个系统完成一次目标追踪需要的时间在 1ms 以内，系统能够达到 1000 帧以上的处理速度。



图 6.19 使用目标追踪测试平台实验结果

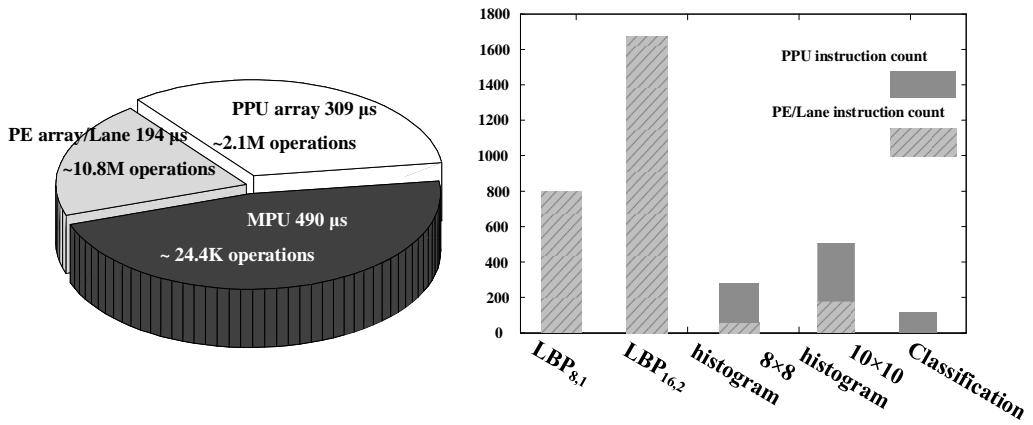


图 6.20 目标追踪算法分析

表 6.3 比较了本文和其他视觉芯片^[32, 72-74]的目标追踪性能。这些报道的视觉芯片使用阈值分割或者通过图像传感器得到二值图像，并在该二值图像的基础上进行追踪。在背景和目标灰度相差很大且光照恒定的情况下这些算法是有效的，但是在背景复杂、光照变化剧烈的情况下这类基于二值图像处理的算法迅速失效，无法完成目标追踪任务。本文利用 LBP 对图像中目标进行纹理描述，实现了具有鲁棒性的目标追踪。

表 6.3 追踪算法以及实现性能对比

	本设计	Komuro[72]	Miao[73]	Zhang[32]	Zhao[74]
Image grayscale	8-bit	2-bit	2-bit	8-bit	8-bit
Algorithm	LBP based	Binary image based	Binary image based	Binary image based	Binary image based
Architecture	PE array PPU array	PE array	PE array	PE array RP array	NA
Frequency	50 MHz	40 MHz	20 MHz	100 MHz	N/A
Tracking Speed	1000 fps	1000 fps	1000 fps	10000 fps	100 fps

6.4 性能比较

前面的测试结果表明，本文提出的基于多级异构并行处理架构的具备图像处理、特征点检测、特征描述以及特征分类等多种视觉处理能力，可通过编程实现多种应用，并能够达到较高的系统性能。该架构具有多种并

行度，其中包括像素级并行，块并行以及分布式并行，同时充分考虑了与现代视觉处理流程的结合，特别是与特征点检测和局部特征表达方法的结合。

在 FAST 角点检测中，本文在一系列实现中达到了最高的处理速度。在进行人脸检测、识别、目标追踪等算法时，本文使用了更具鲁棒性和表达能力的 LBP 特征以及 HoG 特征，其特征维度达到了以往视觉芯片所使用 PPED 特征向量的十倍以上。

表 6.4 本文视觉芯片与其他可编程视觉芯片的对比

Ref	本设计	JSSC2014[31]	JSSC2011[32]	JSSC2009[28]
时钟频率	50MHz	50MHz	100MHz	50MHz
块处理能力	是	无	无	无
并行度	像素级、块级、分布式并行	像素级、行级、分布式并行	像素级、行级	行级
微处理器	32 位 MPU	32 位 MPU	8 位 MPU	32 位 MPU
神经网络处理能力	SOM 神经网络	SOM 神经网络	无	无
神经网络节点数	512	256	无	无
传感器处理器流水	是	无	无	有
可用特征	LBP, HoG, PPED 等	PPED	PPED	NA
向量维度	>600	32	32	NA
分类方式	AdaBoost、SOM	SOM	MPU 串行分类	MIPS 串行分类
分类精度	高	低	低	低
峰值性能	64GOPS	12GOPS	44GOPS	76.8GOPS
系统级帧率	340fps@256×256 (人脸检测)	24fps@256×256 (人脸检测)	10fps@256×256 (人脸检测)	360fps@128×128 (姿态检测)
	1000fps (目标追踪)	无	10,000fps (目标追踪)	无

表 6.4 将 HERO 架构和近年来最为优秀的可编程视觉芯片设计进行了全面对比。HERO 使用多种并行度计算，并且是唯一具备块并行处理能力的架构，因此在局部特征表达、匹配分类上较其他架构都具有巨大的优势。与 JSSC2014 相同，本文同样集成了 SOM 神经网络，但神经网络规模更大，为 512 个 SOM 神经元。JSSC2011 和 JSSC2014 在进行图像描述时，均只能采用 PPED 特征，本文可以则可以采用 LBP, HoG 等多种局部特征描述方法来进行图像表达，所使用的特征向量维度可以达到 600 维度以上，

而 JSSC2011 和 JSSC2014 中 PPED 特征向量维度仅为 32 维。因此，本文在人脸检测、识别等的准确率、目标追踪、多目标检测的精度上都有较大程度的提高。在分类方法上，本文可以使用 SOM 神经网络也可以使用 AdaBoost 机器学习算法，较其他实现方式而言更加灵活。在 50MHz 的主频下，HERO 的峰值运算性能则达到了 64GOPS。最后，我们比较了几款视觉芯片在完成完整应用时的系统性能。对 256×256 大小分辨率的图像进行人脸检测时，HERO 的性能远远超过了 JSSC2014 和 JSSC2011，虽然 JSSC2009 实现了 360fps 的姿态检测算法，但是其处理图像大小仅仅为 128×128 ，因此整体系统性能仍然是弱于 HERO 的。

表 6.5 HERO 与动态可重构混合处理器使用 Arria V FPGA 硬件资源对比

	本设计	JSSC2014[31]
逻辑资源使用率	90%	53%
存储器资源使用率	23%	13%
峰值运算性能	64GOPS	12GOPS

本文在 Arria V FPGA 上实现了 JSSC2014 中的动态可重构混合处理器^[31]，并比较了该处理器和 HERO 处理器的硬件资源的使用情况。从表 6.5 的对比中可以看到，虽然 HERO 在逻辑资源使用率以及存储器资源使用率上超出了文献[31]近 80%，但是其峰值运算性能提升达 5 倍以上，因此本文的实现在硬件效能上明显高于文献[31]。

6.5 小节

本章实现并测试了 HERO 系列架构，搭建了测试平台，设计开发了专用汇编器。在 HERO 系列架构中实现了特征点检测、局部特征描述以及包括人脸检测、识别、目标追踪等在内的复杂应用算法。测试结果表明本文所设计的视觉芯片架构能够完成完整的视觉处理流程，并且达到了较高的系统性能。最后，比较了本文与近年来所报道的其他顶级可编程视觉芯片做了全面的对比，结果表明本文所提出的架构在并行度、图像特征提取能力、分类能力以及最后的综合性能方面都超过了其他视觉芯片。

第7章 总结与展望

7.1 本文的研究工作总结

本论文针对视觉芯片架构、电路以及算法开展了广泛、深入的研究。在分析视觉信息处理框架的基础上提出并实现了 HERO 视觉芯片架构。首次实现了视觉芯片块并行处理，使其不仅能应对传统的图像处理算法，并且能够完成现代计算机视觉算法中特征点检测，特征描述等算法，同时使用分布式并行处理方式实现了特征分类的加速。通过结合块并行处理、分布式并行处理、MIMD 工作模式以及灵活的图像块采样方式，实现了高性能的图像处理、特征提取以及特征分类功能。

本文设计设计了用于并行视觉信息处理的指令集，加强了系统的可编程性和易编程性。通过设计 PPU 电路、Lane 电路以及缓存电路实现了该指令集，利用 Lane 阵列电路完成了 PE 阵列的功能，在没有消耗硬件资源的情况下实现了像素级并行运算。

基于 HERO 架构和电路，本文提出了多种在视觉芯片上并行实现的算法，其中包括 FAST、SIFT、SUSAN 角点检测算法，LBP、HoG 局部特征描述算法，以及分类算法。

最后，本文使用 FPGA 实现了 HERO 架构，并搭建和开发了相应的测试平台和测试软件。在 HERO 系列架构中实现了所设计的特征点检测、局部特征描述以及包括人脸检测、识别、目标追踪等在内的复杂应用算法。测试结果表明本文所设计的视觉芯片架构达到了完成完整的视觉处理流程的目的。在与近年来所报道的其他顶级可编程视觉芯片的性能对比中，本文所提出的架构在并行度、图像特征提取能力、分类能力以及最后的综合性能方法都优于其他视觉芯片。

7.2 工作展望

虽然本文对传统视觉芯片架构进行了大幅度的创新与优化，并且实现了可完成复杂计算机视觉算法的视觉芯片，但是在架构、指令集、电路实现以及算法等方面仍然有很大的提升空间。根据本文的研究成果以及一些近年来关于视觉处理的研究热点，后续的视觉芯片研究可以从以下几个方面开展：

1. 进一步优化系统指令集。本论文结合 MIPS 指令集为 HERO 系列架构开发了一套视觉处理指令，但是，在指令集优化方面的工作仍有不足。我们可以分析在视觉处理算法中最为常用的运算以及消耗指令周期较多的运算，根据这些运算提出专用的指令以提高算法运行的速度。同时，优化指令集更有利于提高视觉芯片可编程性。
2. 进一步优化电路的流水结构，以提高系统性能。目前视觉芯片工作在 100MHz 主频以下，这很大程度上受限于目前的电路设计。为了提高系统工作频率，有必要对视觉芯片电路进行流水化设计。虽然，本文的 Lane 设计已经使用了三级流水线，但是仍然有巨大的优化空间。
3. 针对深度学习方法对视觉芯片处理器架构进行优化。深度学习算法为近年来在人工智能、视觉处理领域具有突破性的研究成果，并且已经在学术界和工业界已经产生了深远的影响。可以预见，在未来数年，深度学习将持续保持研究热度，我们应该把握这次机遇，在理解深度学习算法的基础上对当前视觉芯片架构进行改进。
4. 进一步优化 Cache 设计，尤其是数据 Cache 设计。在视觉处理算法中（尤其是深度神经网络一类算法），数据存取隐含有一定的模式。如果我们能够根据这些模式开发针对视觉处理的缓存就可以极大的提高系统性能。
5. 本文虽然实现了已经具有高鲁棒性的目标追踪，但是，目标模型的建立与更新和目前国际先进的算法还有一定的距离，比如在线学习、多实例学习是目前在视觉芯片中不能实现的，今后的工作可以在朝着方面开展。

事实上，针对上述方向，我们已经开始了初步研究，并取得了一定的进展。我们开展了对深度学习的研究，并且取得了一定的成绩，比如，我们已经使用深度卷积网络（Deep Convolutional Neral Network, DCNN）完成了对目标小车、坦克、不同类型地面的识别，建立了对算法良好的理解。同时，我们发现，只要为 HERO 架构中每个 Lane 增加定点乘法单元并经过小范围的系统优化，我们就可以在 HERO 上用 DCNN 完成识别任务。我们还在进行了面向 DCNN 的 Cache 优化设计。在电路设计方面，我们准备使用更深流水线的 Lane 设计，并且对每个 Lane 实现 SIMD 或者 VLIW 操作。在目标追踪邻域，我们已经搭建了更加完善的测试平台，并进行了一系列的测试，我们在追踪算法上也取得了一定的突破。

总之，视觉芯片是一个充满挑战的交叉研究领域，它的研究不仅涉及半

导体集成电路，图像处理还涉及到计算机体系，计算机视觉，甚至是机器学习和人工智能。视觉芯片的研究任重而道远，吾辈将继续努力，为视觉芯片的发展贡献绵薄之力。

参考文献

- [1] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, pp. 106-154, 1962.
- [2] S. J. Thorpe and M. Fabre-Thorpe, "Seeking categories in the brain," *SCIENCE-NEW YORK THEN WASHINGTON-*, pp. 260-262, 2001.
- [3] D. C. Van Essen and J. L. Gallant, "Neural mechanisms of form and motion processing in the primate visual system," *Neuron*, vol. 13, pp. 1-10, 1994.
- [4] J. G. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision research*, vol. 20, pp. 847-856, 1980.
- [5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91-110, 2004.
- [6] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, pp. 1629-1636, 1990.
- [7] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A CMOS vision chip with SIMD processing element array for 1 ms image processing," in *Solid-State Circuits Conference, 1999. Digest of Technical Papers. ISSCC. 1999 IEEE International*, 1999, pp. 206-207.
- [8] E. Funatsu, Y. Nitta, Y. Miyake, T. Toyoda, J. Ohta, and K. Kyuma, "An artificial retina chip with current-mode focal plane image processing functions," *Electron Devices, IEEE Transactions on*, vol. 44, pp. 1777-1782, 1997.
- [9] H. Kobayashi, J. L. White, and A. A. Abidi, "An active resistor network for Gaussian filtering of images," *Solid-State Circuits, IEEE Journal of*, vol. 26, pp. 738-748, 1991.
- [10] C.-Y. Wu and C.-F. Chiu, "A new structure of the 2-D silicon retina," *Solid-State Circuits, IEEE Journal of*, vol. 30, pp. 890-897, 1995.
- [11] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. Bussat, et al., "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, pp. 699-716, 2014.
- [12] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, et al., "Overview of the spinnaker system architecture," *Computers, IEEE Transactions on*, vol. 62, pp. 2454-2467, 2013.
- [13] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, pp. 668-673, 2014.
- [14] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *Electron Devices, IEEE Transactions on*, vol. 50, pp. 191-199, 2003.
- [15] I. Ishii, K. Yamamoto, and M. Kubozono, "Higher order autocorrelation vision chip," *Electron Devices, IEEE Transactions on*, vol. 53, pp. 1797-1804, 2006.
- [16] F. De Nisi, F. Comper, L. Gonzo, M. Gottardi, D. Stoppa, A. Simoni, et al., "A Novel CMOS Sensor for Position Detection," *IEEE Transactions on Sensors*, vol. 2, pp. 1277-1282, 2003.
- [17] N. Massari, M. Gottardi, L. Gonzo, D. Stoppa, and A. Simoni, "A CMOS image sensor with programmable pixel-level analog processing," *Neural Networks, IEEE Transactions on*, vol. 16, pp. 1673-1684, 2005.

- [18] N. Cottini, M. Gottardi, N. Massari, R. Passerone, and Z. Smilansky, "A 33 W 64 64 Pixel Vision Sensor Embedding Robust Dynamic Background Subtraction for Event Detection and Scene Interpretation," *Solid-State Circuits, IEEE Journal of*, vol. 48, pp. 850-863, 2013.
- [19] B. Zhao, X. Zhang, S. Chen, K.-S. Low, and H. Zhuang, "A 64 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 581-588, 2012.
- [20] Q. Lin, W. Miao, and N. Wu, "A high-speed target tracking CMOS image sensor," in *Solid-State Circuits Conference, 2006. ASSCC 2006. IEEE Asian*, 2006, pp. 139-142.
- [21] W. Miao, Q. Lin, and N. Wu, "A novel vision chip for high-speed target tracking," *Japanese journal of applied physics*, vol. 46, p. 2220, 2007.
- [22] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, et al., "CAVIAR: A 45k neuron, 5M synapse, 12G connects/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking," *Neural Networks, IEEE Transactions on*, vol. 20, pp. 1417-1438, 2009.
- [23] J. A. Leñero-Bardallo, T. Serrano-Gotarredona, and B. Linares-Barranco, "A 3.6 s latency asynchronous frame-free event-driven dynamic-vision-sensor," *Solid-State Circuits, IEEE Journal of*, vol. 46, pp. 1443-1455, 2011.
- [24] C. Posch, T. Serrano-Gotarredona, B. Linares-Barranco, and T. Delbruck, "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output," *Proceedings of the IEEE*, vol. 102, pp. 1470 - 1484, 2014.
- [25] H. Yamashita and C. G. Sodini, "A CMOS imager with a programmable bit-serial column-parallel SIMD/MIMD processor," *Electron Devices, IEEE Transactions on*, vol. 56, pp. 2534-2545, 2009.
- [26] P. Dudek and P. J. Hicks, "A general-purpose processor-per-pixel analog SIMD vision chip," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, pp. 13-20, 2005.
- [27] A. Lopich and P. Dudek, "A SIMD Cellular Processor Array Vision Chip With Asynchronous Processing Capabilities," *IEEE Transactions on Circuits and Systems I-Regular Papers*, vol. 58, pp. 2420-2431, Oct 2011.
- [28] C. C. Cheng, C. H. Lin, C. T. Li, and L. G. Chen, "i Visual: An Intelligent Visual Sensor SoC With 2790 fps CMOS Image Sensor and 205 GOPS/W Vision Processor," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 127-135, Jan 2009.
- [29] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "MATIA: a programmable 80 μ W/frame CMOS block matrix transform imager architecture," *Solid-State Circuits, IEEE Journal of*, vol. 41, pp. 663-672, 2006.
- [30] J. Yang, C. Shi, L. Liu, and N. Wu, "Heterogeneous vision chip and LBP-based algorithm for high-speed tracking," *Electronics Letters*, vol. 50, pp. 438-439, 2014.
- [31] C. Shi, J. Yang, Y. Han, Z. Cao, Q. Qin, L. Liu, et al., "A 1000 fps Vision Chip Based on a Dynamically Reconfigurable Hybrid Architecture Comprising a PE Array Processor and Self-Organizing Map Neural Network," *Solid-State Circuits, IEEE Journal of*, vol. 49, pp. 2067-2082, 2014.
- [32] W. Zhang, Q. Fu, and N.-J. Wu, "A programmable vision chip based on multiple levels of parallel processors," *Solid-State Circuits, IEEE Journal of*, vol. 46, pp. 2132-2147, 2011.
- [33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *arXiv preprint arXiv:1502.01852*, 2015.
- [34] J. Zhang, Y. Yan, and M. Lades, "Face recognition: eigenface, elastic matching, and neural nets," *Proceedings of the IEEE*, vol. 85, pp. 1423-1435, 1997.

- [35] M. Yagi and T. Shibata, "An image representation algorithm compatible with neural-associative-processor-based hardware recognition systems," *Neural Networks, IEEE Transactions on*, vol. 14, pp. 1144-1161, 2003.
- [36] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover.*, 1980.
- [37] C. Harris and M. Stephens, "A combined corner and edge detector," *In Proc. of Fourth Alvey Vision Conference*, pp. 147--151, 1988.
- [38] S. M. Smith and J. M. Brady, "SUSAN - A New Approach to Low Level Image Processing," *International Journal of Computer Vision*, vol. 23, pp. 45--78, 1995.
- [39] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, pp. 346-359, 2008.
- [40] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *In European Conference on Computer Vision*, pp. 430-443, 2006.
- [41] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 971-987, Jul 2002.
- [42] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, pp. 886-893.
- [43] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, pp. 2037-2041, 2006.
- [44] R. Fergus, K. Yu, M. Ranzato, H. Lee, R. Salakhutdinov, and G. Taylor, "Tutorial on deep learning methods for vision," 2012.
- [45] P. Viola and M. J. Jones, "Robust real-time face detection," *International journal of computer vision*, vol. 57, pp. 137-154, 2004.
- [46] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*: Cambridge university press, 2000.
- [47] 韩力群, *人工神经网络理论, 设计及应用*: 化学工业出版社, 2007.
- [48] 石匆, "基于动态可重构混合处理器的高速视觉芯片设计[博士学位论文]," 2014.
- [49] J. Yang, C. Shi, L. Liu, J. Liu, and N. Wu, "Pixel-parallel feature detection on vision chip," *Electronics Letters*, vol. 50, pp. 1839-1841, 2014.
- [50] M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, pp. 1281-1298, 2012.
- [51] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *Computer Vision-ECCV 2012*, ed: Springer, 2012, pp. 864-877.
- [52] D. A. Patterson and J. L. Hennessy, *Computer organization and design: the hardware/software interface*: Newnes, 2013.
- [53] S. M. Smith and J. M. Brady, "SUSAN—a new approach to low level image processing," *International journal of computer vision*, vol. 23, pp. 45-78, 1997.
- [54] E. Rosten, R. Porter, and T. Drummond, "Faster and better: A machine learning approach to corner detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, pp. 105-119, 2010.

- [55] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, ed: Springer, 2006, pp. 430-443.
- [56] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of interest point detectors and feature descriptors for visual tracking," *International journal of computer vision*, vol. 94, pp. 335-360, 2011.
- [57] A. P. Witkin, "Scale-space filtering: A new approach to multi-scale description," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.*, 1984, pp. 150-153.
- [58] T. Lindeberg, "Scale-space theory: A basic tool for analysing structures at different scales," in *Journal of applied statistics*, 1994.
- [59] T. Lindeberg, "Feature detection with automatic scale selection," *International journal of computer vision*, vol. 30, pp. 79-116, 1998.
- [60] J. Clemons, A. Jones, R. Perricone, S. Savarese, and T. Austin, "EFFEX: an embedded processor for computer vision based feature extraction," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, 2011, pp. 1020-1025.
- [61] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, "An 81.6 GOPS object recognition processor based on NoC and visual image processing memory," in *Custom Integrated Circuits Conference, 2007. CICC'07. IEEE*, 2007, pp. 443-446.
- [62] R. Jain, R. Kasturi, and B. G. Schunck, *Machine vision* vol. 5: McGraw-Hill New York, 1995.
- [63] C. Shan, "Learning local binary patterns for gender classification on real-world face images," *Pattern Recognition Letters*, vol. 33, pp. 431-437, 2012.
- [64] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 2006, pp. 1491-1498.
- [65] E. Corvee and F. Bremond, "Body parts detection for people tracking using trees of histogram of oriented gradient descriptors," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 2010, pp. 469-475.
- [66] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464-1480, 1990.
- [67] Altera Arria V. Available: <http://www.altera.com/products/devkits/altera/kit-arria-v-starter.html>
- [68] Y. Moko, Y. Watanabe, T. Komuro, M. Ishikawa, M. Nakajima, and K. Arimoto, "Implementation and evaluation of FAST corner detection on the massively parallel embedded processor MX-G," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, 2011, pp. 157-162.
- [69] K. Dohi, Y. Yorita, Y. Shibata, and K. Oguri, "Pattern compression of fast corner detection for efficient hardware implementation," in *Field programmable logic and applications (FPL), 2011 international conference on*, 2011, pp. 478-481.
- [70] ORL Human Face Library. Available: <https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>
- [71] Labeled Faces in the Wild. Available: <http://vis-www.cs.umass.edu/lfw/>
- [72] T. Komuro, I. Ishii, M. Ishikawa, and A. Yoshida, "A digital vision chip specialized for high-speed target tracking," *Ieee Transactions on Electron Devices*, vol. 50, pp. 191-199, Jan 2003.
- [73] W. Miao, Q. Lin, W. Zhang, and N.-J. Wu, "A programmable SIMD vision chip for real-time vision applications," *Solid-State Circuits, IEEE Journal of*, vol. 43, pp. 1470-1479, 2008.

- [74] B. Zhao, X. Y. Zhang, S. S. Chen, K. S. Low, and H. L. Zhuang, "A 64 x 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization," *Ieee Transactions on Circuits and Systems for Video Technology*, vol. 22, pp. 581-588, Apr 2012.

致谢

毛泽东主席有云“多少事，从来急，天地转，光阴迫，一万年太久，只争朝夕”，这句话亦是我求学五年的感慨，研究虽充满困难与挑战，但带给我更多的是对目标的执着与满腔的热情。回首过去数年时光，我得到了老师、同学、朋友和家人的无私帮助。在此，我希望向他们表达我衷心的感谢。

首先要感谢的是我的两位恩师刘剑研究员和吴南健研究员。我的所有研究工作都是在两位老师的悉心指导下完成的。两位老师深厚的学术造诣和学术思想令我深感钦佩。在攻读博士学位阶段，两位老师总是能够游刃有余的解决我遇到的困惑，指出我研究的不足，并以高屋建瓴的眼光指引我前进的方向。同时两位老师在生活上对我的关心和帮助也让我感到深深的温暖。

感谢与我一起进行视觉芯片研究的石勿师兄。感谢杨永兴、陈哲、李搏、李鸿龙、张忠星、维克多、罗迁师弟，希望你们能够将视觉芯片的研究推向新的高度！

感谢其他老师和同学，他们是：刘力源、冯鹏、窦润江、秦琦、李国峰、楼文峰、刘威扬、周扬帆、韩烨、曹中祥、李全良、刘晓东、陈晶晶、于双铭、底衫、秦晋一、刘朝阳、张钊、郭志强、姚兵兵、曹静、杨锦城、姚圆圆。感谢你们在学习和生活上的帮助和关心！

感谢中国科学院大学、中国科学院半导体研究所对我的培养，感谢各位老师的关心、帮助和指导！

最后要特别感谢我的父亲、母亲、姐姐和其他亲人给我无比的关怀和支持，是你们教会我如何做人、如何快乐地生活，你们是我最大的精神动力，我爱你们！

个人简历、在学期间发表的学术论文与研究成果

个人简历

1987 年 12 月 17 日生于重庆市丰都县。

2006 年 9 月进入天津大学电子与信息工程学院攻读电子科学与技术专业，2010 年 7 月毕业并获得学士学位，同年 9 月进入中国科学院半导体研究所攻读博士学位。

发表的学术论文

- [1] **Jie Yang**, Cong Shi, Liyuan Liu, et al. Heterogeneous vision chip and LBP-based algorithm for high-speed tracking. *Electronics Letters*, 2014, 50(6): 438-439. (SCI 收录)
- [2] **Jie Yang**, Cong Shi, Liyuan Liu, et al. Pixel-parallel feature detection on vision chip. *Electronics Letters*, 2014, 50(24): 1839-1841. (SCI 收录)
- [3] **Jie Yang**, Cong Shi, Zhongxiang Cao, et al. Smart Image Sensing System. *2013 IEEE Sensors Conference*, 2013: 1-4. (EI 收录)
- [4] Cong Shi, **Jie Yang**, Ye Han, et al. A 1000 fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array and self-organizing map neural network. *IEEE Journal of Solid-State Circuits*, 2014, 49(9): 2067-2082. (SCI 收录)
- [5] Cong Shi, **Jie Yang**, Ye Han, et al. A 1000fps vision chip based on a dynamically reconfigurable hybrid architecture comprising a PE array and self-organizing map neural network. *2014 IEEE International Solid-State Circuits Conference*, 2014: 128-129. (EI 收录)
- [6] Cong Shi, **Jie Yang**, Liyuan Liu, et al. A massively parallel keypoint detection and description (MP-KDD) algorithm for high-speed vision chip. *Science China Information Sciences*, 2014, 57(10): 1-12. (SCI 收录)
- [7] Cong Shi, **Jie Yang**, Nanjian Wu, et al. A high speed multi-level-parallel array processor for vision chips. *Science China Information Sciences*, 2014, 57(6): 1-12. (SCI 收录)
- [8] Bo Li, **Jie Yang**, Yongxing Yang, et al. A high-speed vision processor for chip package visual inspection. *2014 IEEE 12th International Conference on Solid-State and Integrated Circuit Technology*, 2014. (EI 收录)

- [9] Zhe Chen, **Jie Yang**, Cong Shi, et al. A novel architecture of local memory for programmable SIMD vision chip. 2013 10th International Conference on ASIC (ASICON), 2013: 1-4. (EI 收录)
- [10] Cong Shi, Zhe Chen, **Jie Yang**, et al. A compact PE memory for vision chip. Journal of Semiconductors, 2014, 35(9). (EI 收录)