目前在汽车应用、机器人、娱乐等方面都需要实时处理图像识别的嵌入式应用，嵌入式应用需要在系统范围内实现实时处理迭代能力和大小，为此，嵌入式处理器已设计出适合图像识别的方法，提出用于某些图像识别算法的专用硬件，以及基于DSP的实时系统增强了图像识别功能等。

1. 关于ARMv8架构的发展与研究：

ARMv8架构[[[1]](#endnote-1)]是ARM公司于2011年发布的新一代处理器架构，是ARM公司首款支持64位指令集的处理器架构，目前ARMv8架构的处理器应用广泛，鲲鹏920处理器即是基于ARMv8架构的，而目前已经进行过多项基于ARMv8架构方面的研究。

国防科技大学姜浩等人研究的面向ARMv8 64位多核处理器的QGEMM设计与实现[[[2]](#endnote-2)]，采用计算指令顺序优化调度和寄存器轮转策略，回避不必要的规格化操作，增加指令级并行度，提升了算法效率，为之后进行OpenCV内嵌汇编优化方法的探索提供了一定算法上的借鉴意义。

陈暾等人基于ARMv8平台进行了多维FFT的实现与优化研究[[[3]](#endnote-3)]，实现多维FFT变换的基础上，提出了蝶形网络优化、蝶形计算优化、蝶形自动生成、SIMD优化、内存对齐、cache-aware的分块算法和高效转置算法等优化方法，为 ARMv8平台上程序优化提供了新的思路。

曾庆淼基于ARMv8进行高吞吐率口令破解关键技术研究[[[4]](#endnote-4)]，利用ARMv8架构优秀的计算性能功耗比、性能价格比，在高性能计算领域发挥强有力作用。

（二）关于鲲鹏920处理器的发展与研究：

上海交通大学与美国普林斯顿大学相关研究者研究了鲲鹏系列中的鲲鹏916处理器[[[5]](#endnote-5)]，针对鲲鹏916在HPC工作负载方面的潜力采用三个基准、三个内核、三个微型应用程序、一个实际应用进行系统评估，比较鲲鹏916和Intel XeonE5-2680v3/4(Haswell/Broadwell)的性能结果，结果表明鲲鹏916具有比两个Intel处理器更高的内存带宽，可以为运行内存绑定HPC应用程序提供更好的性能。

而2019年1月发布的鲲鹏920处理器，在性能上是鲲鹏916的三倍，内存带宽提升2,4倍，高度集成CPU、南桥、网卡、SAS存储控制器等4颗芯片的功能，能够释放出服务器更多槽位，用于扩展更多加速部件功能，大幅提高系统的集成度。相对于鲲鹏916处理器，鲲鹏920处理器在相同功耗下性能表现提高了35%。

为更好实现软件移植，华为公司还推出了鲲鹏开发套件(包括扫描评估工具Dependency Advisor, 迁移工具Porting Advisor, 和性能优化工具Tuning Kit）以帮助用户评估工程的可移植性，做软件的移植，以及做基于鲲鹏平台的性能加速。

（三）关于OpenCV移植与函数优化方面[[[6]](#endnote-6)]：

中国科学技术大学的陈思润、顾乃杰等人分析并研究ARM架构硬件特性，使用Cortex-A系列支持的SIMD数据级并行计算技术NEON对OpenCV函数库中的滤波函数进行优化[[[7]](#endnote-7)]。实验结果表明，中值滤波函数优化效果显著，对比OpenCV源码性能提升了17倍，图像处理模块中的其他滤波函数均有较大的性能提升。为结合SIMD、NEON技术研究OpenCV在鲲鹏920上的优化提供借鉴意义。

国防科技大学的孙广辉、扈啸、王蕊在Linux系统下，以FT-M7002为平台，通过对OpenCV进行交叉编译，实现OpenCV在FT-M7002上的移植的一般方法，结合FT-M7002体系结构支持DMA和cache操作与向量化的编程相结合，实现了OpenCV中的add、addWeighted、subtract和multiply函数在FT-M7002上的向量化，并且总结出针对OpenCV底层对像素做矩阵运算处理的函数进行向量化改造的一般方法。依据向量化改造结果，使OpenCV在FT-M7002的单核运行性能最多可提升13.7倍[[[8]](#endnote-8)]。

李津等人在 TI6678平台上实现了OpenCV 的移植[[[9]](#endnote-9)]，深入分析了一类OpenCV库函数在 TI6678硬件平台运行的计算特征和数据流，提出了一种针对这类 OpenCV库函数的优化方法，将 TI6678体系结构支持的DMA 和Cache操作与 OpenCV并行框架高效结合，实现这类 OpenCV库函数在 TI6678芯片上的优化和多核并行。

有关学者为有效改善实际应用中OpenCV受来自外部处理器等方面的限制，例如存储类宽带的限制，基于Xilinx的HLS作为OpenCV性能优化的着手点[[[10]](#endnote-10)]，对OpenCV函数以及图像处理的设计原理、性能优势展开研究。还有一些学者基于ARM11处理器[[[11]](#endnote-11)]、ARM9处理器[[[12]](#endnote-12)]，以Linux为软件平台，通过移植OpenCV，进行嵌入式研究，构建了具有计算机视觉功能的嵌入式系统的解决方案；除此之外还有基于测量的多核处理器的OpenCV功率优化技术研究[[[13]](#endnote-13)]，在异构多核处理器的基础上，提出了一种基于测量的开源计算机视觉应用程序库OpenCV优化技术。

Coombs J和Prabhu R研究说明了关于使用C实现、内存约束、浮点支持和使用供应商优化的库和集成加速器或协同处理器去最大化性能，并介绍德州仪器(TI)通过在C6000™数字信号处理器(D SP)架构上运行OpenCV来优化视觉系统，建立基准比较DSP ARM®片上系统(SOC)处理器与仅ARM的设备的性能[[[14]](#endnote-14)]。

Sugano H等人研究了基于蜂窝引擎处理器优化的OpenCV实现[[[15]](#endnote-15)]，在Cell Broad-上并行执行OpenCV库带引擎（Cell），实验结果表明，大多数功能针对Cell处理器进行了优化的速度更快。

（四）关于ARM架构的交叉编译、应用移植等技术方面：

兰州大学的戈孜荣分析对比了x86和ARM架构在指令集、体系结构方面的差异性，并概括了ARMv8-A架构的特性以及其处理器的特点，完成借助华为鲲鹏移植套件完GROMACS从x86到ARM的完整移植过程并研究了面向ARM架构在硬件和软件层面的优化方法[[[16]](#endnote-16)]。

张欢庆等人介绍了一种利用crosstool构建基于ARM的嵌入式Linux的交叉编译环境的方法[[[17]](#endnote-17)]，通过使用开源项目Sky-Eye模拟ARM硬件环境进行仿真验证测试，实验结果也表明能够使用x86-ARM的交叉编译工具进行ARM平台上的嵌入式系统开发，具有可靠的稳定性。

关于ARM的像素处理技术方面的发展现状：SIMD（单个指令、多个数据）处理对于像素数据特别有用，相同指令可同时在多个像素上使用。SSE 是英特尔的 SIMD 技术，它存在于所有现代 x86 芯片上。ARM 具有类似的技术——NEON，它是 Cortex A9 中的可选协处理器。NEON 可以同时处理多达8个像素，有时可处理16个像素，而 CPU 一次只能处理一个元素。这对计算机视觉开发人员非常有吸引力，因为通常很容易获得三到四倍的性能加速，并经过仔细的优化甚至超过六倍[[[18]](#endnote-18)]。

（五）结合ARM与OpenCV的应用：

基于OpenCV的ARM嵌入式网络视频监控系统[[[19]](#endnote-19)]，将OpenCV函数库移植到ARM-Linux嵌入式系统中，并基于OpenCV函数库和Surendra算法开发与实现了一个嵌 入 式 网 络 视 频 监 控 系 统 。

行人保护系统是先进驾驶员辅助系统(ADAS)中避免道路事故的快速发展的系统之一。 利用友好的ARM板S3C2440和Haar级联和正面人脸检测算法，在OpenCV平台上开发了高效、有效的方法[[[20]](#endnote-20)]。

1. [] ARM Architecture Reference Manual.pdf（ARMv8, for ARMv8-A architecture profile） [↑](#endnote-ref-1)
2. [] 姜浩,杜琦,郭敏,全哲,左克,王锋,杨灿群.面向ARMv8 64位多核处理器的QGEMM设计与实现[J].计算机学报,2017,40(09):2018-2029. [↑](#endnote-ref-2)
3. [] 陈暾,李志豪,贾海鹏,张云泉.基于ARMv8平台的多维FFT实现与优化研究[J].计算机学报,2019,42(11):2384-2402. [↑](#endnote-ref-3)
4. [] 曾庆淼.基于ARMv8的高吞吐率口令破解关键技术研究[D].华南理工大学,2019. [↑](#endnote-ref-4)
5. [] Wang Y C, Chen J K, Li B R, et al. An Empirical Study of HPC Workloads on Huawei Kunpeng 916 Processor[C]//2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2019: 360-367. [↑](#endnote-ref-5)
6. [] [（美）柯尔特·甘瑟尔罗斯（Kurt Guntheroth）](https://search-e.jd.com/searchDigitalBook?ajaxSearch=0&key=%EF%BC%88%E7%BE%8E%EF%BC%89%E6%9F%AF%E5%B0%94%E7%89%B9%C2%B7%E7%94%98%E7%91%9F%E5%B0%94%E7%BD%97%E6%96%AF%EF%BC%88Kurt%20Guntheroth%EF%BC%89)著 [杨文轩](https://search-e.jd.com/searchDigitalBook?ajaxSearch=0&key=%E6%9D%A8%E6%96%87%E8%BD%A9) 译C++性能优化指南. 人民邮电出版社. 2018. [↑](#endnote-ref-6)
7. [] 陈思润,顾乃杰,苏俊杰,贺爱香.基于ARM架构的滤波函数优化[J].计算机应用与软件,2018,35(09):138-143. [↑](#endnote-ref-7)
8. [] 孙广辉,扈啸,王蕊. 基于FT-M7002的OpenCV移植与优化[A]. 中国计算机学会.第二十二届计算机工程与工艺年会暨第八届微处理器技术论坛论文集[C].中国计算机学会:中国计算机学会计算机工程与工艺专业委员会,2018:9. [↑](#endnote-ref-8)
9. []李津,罗昕颉,扈啸,陈跃跃.基于TI 6678多核DSP的OpenCV并行优化[J].计算机工程与科学,2018,40(05):780-786. [↑](#endnote-ref-9)
10. []赵睿.基于Xilinx的HLS实现OpenCV的研究与设计[J].科技创新与应用,2016(21):56. [↑](#endnote-ref-10)
11. []陈鹏. 基于ARM11的嵌入式人脸识别系统的设计和实现[D].郑州大学,2013. [↑](#endnote-ref-11)
12. []迟颖. 基于ARM和OpenCV的视频监控系统设计[D].大连海事大学,2016. [↑](#endnote-ref-12)
13. [] Jung H, Koo K, Yang H. Measurement-Based Power Optimization Technique for OpenCV on Heterogeneous Multicore Processor[J]. Symmetry, 2019, 11(12): 1488. [↑](#endnote-ref-13)
14. [] Coombs J, Prabhu R. OpenCV on TI’s DSP+ ARM® platforms: Mitigating the challenges of porting OpenCV to embedded platforms[J]. Texas Instruments, 2011. [↑](#endnote-ref-14)
15. [] Sugano H, Miyamoto R. Opencv implementation optimized for a cell broadband engine processor[C]//2009 IEEE 13th Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop. IEEE, 2009: 182-187. [↑](#endnote-ref-15)
16. [] 戈孜荣. 面向ARM架构的HPC系统性能评测及应用移植优化[D].兰州大学,2020. [↑](#endnote-ref-16)
17. [] 张欢庆,高丽,宋承祥.基于ARM的嵌入式Linux交叉编译环境的研究与实现[J].计算机与数字工程,2012,40(02):151-153. [↑](#endnote-ref-17)
18. [] Pulli K, Baksheev A, Kornyakov K, et al. Real-time computer vision with OpenCV[J]. Communications of the ACM, 2012, 55(6): 61-69. [↑](#endnote-ref-18)
19. [] 梁艳.基于OpenCV的ARM嵌入式网络视频监控系统[J].微型机与应用,2013,32(09):29-31. [↑](#endnote-ref-19)
20. [] Dixit R S, Gandhe S T. Pedestrian detection system for ADAS using Friendly ARM[C]//2015 International Conference on Energy Systems and Applications. IEEE, 2015: 557-560. [↑](#endnote-ref-20)