. Materia: Temas Selectos de Electrónica
Profesor: M. en C. Arturo Ocampo Álvarez

Antecedentes.

El don de la vista ... Esta habilidad increíble trae a nuestro mundo un vivo enfoque, colorido y nos permite comprender e interactuar con nuestro entorno. Nuestros sentidos visuales amplían nuestra conciencia al mundo más allá de nuestro alcance, mientras se procesan grandes cantidades de información para tomar decisiones rápidas sobre lo que estamos viendo.

Si bien la investigación de visión por computador ha generado numerosos libros y artículos de investigación, hay muy poca información disponible para guiar a los diseñadores de sistemas en el diseño de soluciones prácticas de visión por ordenador, incluyendo la selección de chips, cámaras, algoritmos, herramientas y lenguajes de programación. Así que, unas de las metas iniciales de esta investigación ha sido averiguar que plataformas de hardware existentes en el mercado pueden servir para realizar sistemas de visión embebido.

En el desarrollo de la visión por computador o aplicaciones de procesamiento de imágenes es común una computadora de escritorio, y las opciones de plataformas de hardware son muy simples: si no se requiere mucho procesamiento, entonces se puede utiliza un ordenador portátil barato o un PC con GPU integrada, o si necesitamos una gran cantidad de poder de procesamiento se utiliza una CPU rápida con una potente tarjeta GPU dedicada. Pero para sistemas embebidos, hay muchas más opciones para elegir, y no hay un solo dispositivo que es adecuado para todas las plataformas embebidas porque cada uno tiene algunas ventajas y algunas desventajas. Así que este es un resumen básico de hardware integrado que se puede utilizar para la visión por computador y procesamiento de imágenes:

Dispositivo/Familia	Cálculos GFLOPS	Batería Watts	Características
Microcontroladores	<0.2	<0.3	La mayoría de los microcontroladores (por ejemplo: Arduino, AVR, PIC) son demasiado lentos para el procesamiento de la cámara, pero un ARM de 32 bits Cortex-M4 (por ejemplo: $15 168MHz STM32F407) pueden manejar algunas aplicaciones muy básicas de la cámara tales como seguimiento y detección de color. Los microcontroladores sólo admiten sistemas operativos muy mínimos, por lo que normalmente el software se ejecuta como el firmware de bajo nivel, y se debe escribir la mayoría de los algoritmos, con la ventaja de acceso directo al hardware como pines I/O y temporizadores y funcionamiento en tiempo real y diseño eléctrico muy simple. Para los sistemas de visión con microcontrolador que ya están disponibles, se puede optar por $80 CMUcam o el próximo $55 OpenMV bordo.
Dispositivo movil SOC Targeta o Tablet	1-25	1-6	La CPU ARM de un dispositivo móvil puede proporcionar tanto una gran velocidad como un bajo consumo de energía. La mayoría de las tarjetas de propósito general hacen el procesamiento con cálculos de números enteros, y son de un precio accesible como el Cortex-A8 (por ejemplo: $45 BEAGLEBONE Black) o incluso un ARM11 (por ejemplo: $25 RASPBERRY Pi) podría ser lo suficientemente bueno, pero si es necesario manejar operaciones con punto flotante, entonces definitivamente se necesita un Cortex-A9 (por ejemplo: $65 quad-core ODROID-U3) o Cortex-A15 (por ejemplo: $139 de cuatro núcleos ODROID-XU o una tarjeta de cuatro núcleos $192 Jetson

			TK1) ya que su hardware contiene FPU que es mucho más rápido que el FPU del Cortex-A8. Si se requiere el mejor rendimiento disponible en esta clase de tarjetas entonces la aceleración de GPU del $192 Jetson TK1 es más rápido, o si es necesario un tamaño más pequeño una Gumstix Overo (> $ 200) es muy útil, pero si se desea eficiencia en la CPU entonces un Cortex A7 (por ejemplo: $55 A20-OLinuXino-LIME2) es la CPU más apropiada. Pero si también es necesario visualizar imágenes en la pantalla entonces una tableta con Android o Linux podría ser una mejor opción (mediante Wifi o Bluetooth a un microcontrolador si requerimos E/S de acceso). El desarrollo de software para un SOC ARM es similar al software de escritorio y bibliotecas como OpenCV son compatibles con ARM, pero no es tan fácil como x86. La ejecución de Android o un Linux estándar como su sistema operativo significa que su código puede ser interrumpido al igual que en un PC, pero algunos sistemas operativos en tiempo real están disponibles.
Diseño de hardware con FPGA	50-1000	1-3	Trabajar con FPGAs (por ejemplo: $199 Kit Cyclone II + $85 una cámara de 5MP) el procesamiento pueden ser extremadamente rápido con el mínimo de consumo en la batería, pero son muy complejos para diseñar, potencialmente tomaría meses o años! Con los FPGAs se diseña la lógica de hardware, no el software, por lo que el modelo de programación es completamente diferente a la programación de software, aunque se puede insertar una CPU y un FPGA (por ejemplo: sbRIO-9637, este producto esta disponible solo en cantidades OEM).

La relación entre el software OpenCV y el Hardware.

Introducción.

OpenCV es una biblioteca de visión por ordenador de distribución libre y de código abierto que ofrece una amplia gama de funcionalidad bajo la licencia permisiva de Berkeley Software Distribution (BSD). La biblioteca en sí está escrito en C++ y también es utilizada a través de C o del lenguaje Python. Miles de desarrolladores utilizan OpenCV para alimentar sus propias aplicaciones especializadas, por lo que es la biblioteca más ampliamente utilizada. El proyecto OpenCV está en desarrollo activo, con regularidad actualizaciones para eliminar errores y añadir nuevas funcionalidades. Los objetivos del esfuerzo de desarrollo se aplican a la arquitectura x86 y utiliza el Rendimiento Integrado como una propiedad Primitiva de Intel (IPP). Un comunicado reciente también ha añadido soporte para la unidad de procesamiento gráfico (GPU) en la arquitectura de cálculo paralelo CUDA de NVIDIA.

El mayor atractivo de OpenCV es la enorme amplitud de los algoritmos incluidos en su distribución estándar. Estos van desde el uso de aplicaciones de bajo nivel, filtrado de imágenes y transformación de sofisticados análisis de características y funcionalidad de aprendizaje automático. Para muchos desarrolladores, OpenCV representa una atractiva caja de herramientas completa de algoritmos útiles, bien probados que puede servir como bloques para sus propias aplicaciones especializadas. *La pregunta entonces es si o no OpenCV se puede utilizar directamente en sus sistemas embebidos.*

A pesar de su enfoque de desarrollo original para su uso con las estaciones de trabajo de PC, OpenCV también puede ser una herramienta útil para el desarrollo de sistemas de visión embebida. Hay bibliotecas en lenguaje C de varios proveedores que ofrecen capacidades de OpenCV como en varios

sistemas embebidos, pero pocos pueden igualar la ubicuidad de OpenCV en el campo de la visión por computador o la gran amplitud de sus algoritmos incluidos.

OpenCV ya ha sido portado a la arquitectura ARM, que como ya se observo en la tabla anterior es una opción muy popular para procesadores embebidos. Ciertamente es posible hacer una compilación cruzada usando el código de OpenCV, pero las limitaciones de memoria y otros consideraciones arquitectónicas pueden suponer un problema. Esta investigación examina algunos de los obstáculos específicos que deben superarse para poder usar OpenCV y lograr un rendimiento aceptable en una tarjeta de propósito general. Finalmente, el documento describe una nueva opción desarrollada por Texas Instruments (TI) para llevar el OpenCV a su procesador de digital señales (DSP) C6000.

Típico sistema de visión embebida.

El crecimiento continuo de aplicaciones de visión embebida coloca demandas contradictorias sobre los desarrolladores de sistemas embebidos. Cada vez más sofisticados algoritmos de visión requieren más memoria y capacidad de procesamiento, pero el precio y restricciones de despliegue requieren dispositivos que cuesten menos dinero y consuman menos energía.

Vamos a empezar con las aplicaciones de visión industrial. Una tarea común en la visión industrial es la inspección de montajes, para detectar y clasificar los tipos de objetos para maximizar la velocidad de fabricación y calidad. Estos algoritmos de visión son a menudo costosos y se ejecutan en estaciones de trabajo; la migración a un DSP es una manera de ahorrar en precio y consumo de energía. Incluso las aplicaciones que ya se implementan con sistemas embebidos se pueden mejorar, por ejemplo, muchos sistemas de visión industrial comparten la forma básica que se ilustra en la Figura 1. El procesador de señal de imagen (ISP) es un FPGA programable que realiza el pre-procesamiento en tiempo crítico antes del procesamiento en el DSP. Esta FPGA vuelve al sistema más caro y consume más energía y es proporcional a su carga de trabajo. Por lo tanto, se pretende que el sistema de visión maximizar la eficiencia del sistema embebido integrando la mayor cantidad de pre-procesamiento como sea posible en el DSP. El desafío entonces es mantener el ritmo haciendo el reconocimiento de objetos en el DSP.

Figura 1. Sistema típico de Visión.

Los istemas de próxima generación deben procesar más datos en menos tiempo para dar cabida a una mejor resolución de la cámara y velocidad de fotogramas, así como velocidades de línea de montaje más rápidos.

Aplicaciones de vigilancia de vídeo proporcionan otra perspectiva sobre la evolución de la visión integrada. Los sistemas tradicional de vigilancia están menos preocupados por el análisis de la visión y se enfocan más en codificar el registro de datos de vídeo. Sin embargo, como los algoritmos de visión se están mejorando, la videovigilancia incorporará más monitoreo automatizado y análisis de los datos. Los ejemplos van desde el movimiento y manipulación de la cámara, detección y conteo de personas hasta lectura de matrículas. Estos algoritmos permiten la transmisión de metadatos, o la creación de los registros automatizados de actividad Como los algoritmos de visión se vuelven cada ves más capaces y confiables, los sistemas de vigilancia de

vídeo serán más automatizados y sofisticados. Esto presenta un desafío particular para los sistemas de vigilancia de vídeo integrados, ya que los algoritmos de vanguardia que se desarrollan en las PC pueden requerir una mayor optimización para funcionar eficientemente en un dispositivo embebido.

Un último ejemplo de aplicación de la amplia categoría de visión embebida es la visión de la automoción. Muchos sistemas de visión para automóviles pueden ser reducidos a un diagrama de bloques similar a la figura 1, que consiste esencialmente en una cámara, un FPGA para el preprocesamiento y un DSP para aplicar algoritmos de visión intensivos.

La fiabilidad es la principal preocupación en aplicaciones tales como la alerta de cambio de carril, la asistencia de dirección y detección de proximidad. Los algoritmos de visión utilizados en la visión de la automoción están bajo constante desarrollo y el uso de software de alto nivel en un PC, simplemente no es una opción. La transición desde el PC al DSP es un paso crítico en el desarrollo de aplicaciones de visión de automoción. La escritura y reescritura de algoritmos para lograr rendimiento en tiempo real aceptable es un foco importante de desarrollo. Esto sólo se consigue con sistemas embebidos que se vuelven cada día más sofisticados, incorporando múltiples entradas de cámara y múltiples núcleos de procesamiento.

Un eficiente software en el DSP desempeña un papel fundamental en todas las aplicaciones de visión embebida. La posibilidad de utilizar software de alto nivel como OpenCV para facilitar el desarrollo de algoritmos de manera rápida es atractivo, pero la optimización de que el software para una nueva plataforma es un escollo importante. Por el contrario, el logro de un rendimiento aceptable optimizando un software para el DSP es simplemente poco realista. En la siguiente sección de este informe, se consideran los desafíos clave asociados con la portabilidad y la optimización de la biblioteca OpenCV para ejecutarse en un dispositivo embebido.

Desafíos para lograr la portabilidad de OpenCV en un dispositivo embebido.

OpenCV es de código abierto y escrito enteramente en C/C++, la biblioteca ha sido compilada y portada como a una variedad de plataformas. Sin embargo, sólo la reconstrucción de la biblioteca en una plataforma embebida puede no dar el rendimiento en tiempo real exigió. Al mismo tiempo, la reescritura y optimización de forma manual de la toda la biblioteca OpenCV, para una nueva arquitectura representa una enorme cantidad de trabajo. Los compiladores apropiados son críticos para solucionar estos desafíos. El compilador GCC se ha utilizado en OpenCV con éxito para la plataformas ARM, pero GCC no está disponible en arquitecturas más especializados como los DSP. Estos dispositivos normalmente se basan en los compiladores propietarios que no son tan compatible con los estándares. Estos compiladores pueden tener un fuerte enfoque hacia el lenguaje C y ser menos capaces de optimizar el código C++. La versión actual de OpenCV se basa principalmente en C++ Standard Template Library (STL), así como del GCC y C99, que no están bien apoyados para ciertos compiladores de sistemas embebidos. Por estas razones, puede ser necesario volver a OpenCV versión 1.1 o anterior las cuales están escritos casi en su totalidad en C. El código fuente de OpenCV incluye muchas optimizaciones de bajo nivel para procesadores x86 que no son aplicables a las plataformas ARM o DSP. Estas optimizaciones se pueden reemplazar con las bibliotecas de apoyo proporcionados por el proveedor o funciones intrínsecas que hacen uso explícito de la arquitectura específica para manipular las instrucción simples de múltiples datos (SIMD) aceleraran la ejecución del código.

Interfaces de programación de aplicaciones (API) para OpenCV a menudo permiten que los datos que se deben proporcionar en múltiples formatos, hacen que se puedan complicar y optimizar las funciones para un nuevo dispositivo. Limitando estas funciones a un solo tipo de datos o del mismo modo, inlining pequeñas funciones internas, de uso frecuente puede proporcionar un alto rendimiento para funciones de visión de alto

nivel. La mayoría de los proveedores proporcionan las bibliotecas optimizadas para proporcionar el mejor rendimiento en el dispositivo para realizar las operaciones matemáticas de bajo nivel, en la imagen y la funcionalidad de visión.
Texas Instruments (TI) es una de las pocas empresas que proporciona la visión de imágenes y bibliotecas que pueden reemplazar una porción del código para una función de OpenCV, en algunos casos, toda la función. Del mismo modo, las bibliotecas de procesamiento matemático y la optimización de señales también puede dar un impulso significativo para maximizar el potencial de las funciones de OpenCV en dispositivos embebidos. El uso de estas bibliotecas optimizadas debajo de las API en OpenCV pueden maximizar el rendimiento mediante la utilización de la arquitectura específica, manteniendo la interfaz estándar del software de alto nivel. En otras palabras, estas bibliotecas de bajo nivel pueden acelerar las funciones de OpenCV sin romper el código preexistente de la aplicación que está escrito utilizando las API de OpenCV estándar. Otro de los retos que a menudo se enfrentan al usar funciones OpenCV en un entorno de procesamiento embebido es la falta de soporte nativo para operaciones matemáticas de puntos flotante. Esto plantea un problema significativo para usar OpenCV, ya que incluye una serie de funciones especializadas para el procesamiento de imágenes que dependen en gran medida del cálculo en puntos flotante. OpenCV es compatible con una amplia gama de tipos de datos de imagen, incluyendo representaciones de punto fijo y punto flotante. Sin embargo, algunas funciones especializadas para la transformación de la imagen siempre utilizan operaciones matemáticas de punto flotante, independientemente del original tipo de datos de imagen. Estos algoritmos intensivos requieren soporte nativo para conseguir un rendimiento en tiempo real en una aplicación embebida. En la figura 2 se compara el rendimiento de varias funciones de OpenCV que dependen del puntos flotante para el procesamiento en varias tarjetas. El procesador ARM9 utilizado carece de soporte de punto flotante nativo, mientras que el procesador ARM Cortex-A8 incluye el dispositivo NEON que sirve de apoyo operaciones matemáticas y ofrece un aumento del doble en el rendimiento. También se incluye el DSP C674x de TI que realiza operaciones en punto flotante, que está muy optimizado para el cálculo intensivo y ofrece un mayor soporte.

Function Name	ARM9™ (ms)	ARM Cortex-A8 (ms)	C674x DSP (ms)
cvCornerEigenValsandVecs	4746	2655	402
cvGoodFeaturestoTrack	2040	1234	268
cvWarpAffine	82	37	17
cvOpticalFlowPyrLK	9560	5240	344
cvMulSpecturm	104	69	11
cvHaarDetectObject	17500	8217	1180

Figura 2. Rendimiento de las funciones de OpenCV con punto flotante. Tamaño de imagen de 320 × 240; todas las tarjetas tienen núcleos que funcionan a 300 MHz; los núcleos ARM9 y DSP C674x son probados usando la tarjeta OMAP-L138 que integra DSP + ARM; el núcleo ARM Cortex-A8 es probado usando la tarjeta DaVinci DM3730.

Además de la arquitectura del procesador, también puede haber restricciones de memoria y los requisitos especiales para una operación en tiempo real determinista. Los dispositivos multinúcleo también son cada vez más comunes, y la utilización de estos núcleos son más eficientemente para maximizar el rendimiento pero trae sus propios desafíos. Los dispositivos multinúcleo pueden consistir en núcleos homogéneos, como los dispositivos de doble ARM, o pueden integrar un ARM con un núcleo heterogéneo tal como un DSP o GPU.

Los dispositivos SOC también integran periféricos y aceleradores para reducir la complejidad general del sistema. Muchas funciones de OpenCV pueden beneficiarse enormemente de la utilización de estos núcleos de procesamiento especializados así como del uso de punto flotante. Los algoritmos que son altamente paralelizables pueden ser una buena opción para una GPU integrada. Los algoritmos de procesamiento de imágenes que no están paralelizados con facilidad, pero todavía requieren operaciones intensivas en punto flotante puede ser más adecuados para un núcleo DSP. Las funciones de preprocesamiento de bajo nivel, como las conversiones de espacio de color, la reducción del ruido y los cálculos estadísticos suelen estar bien adaptado al hardware de propósito único como un FPGA o las aplicaciones específicas en circuitos integrados (ASIC).

Los dispositivos integrados que permiten a los desarrolladores dividir efectivamente su aplicación, incluyendo OpenCV, entre los componentes heterogéneos más adecuados pueden realizar un mejor desempeño. Efectivamente utilizando y compartiendo memoria del dispositivo es uno de los principales desafíos en materia de desarrollo integrado. Tanto la memoria de acceso aleatorio (RAM) y memoria de sólo lectura (ROM) son escasos, las aplicaciones debe hacer un uso prudente de estos recursos. Muchas aplicaciones de hoy en día requieren un sistema operativo completo (OS), lo que hace que el manejo de la memoria en el dispositivo sea aún más crítica. Una aplicación de visión embebida usando OpenCV necesita razonablemente gran capacidad de memoria con un ancho de banda suficiente y un rápido tiempo de acceso.