RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

REPORTE TÉCNICO
Reconocimiento de Patrones

Estado actual de los algoritmos de reconocimiento de rostro usando tecnología GPU

Ing. Danis López Naranjo,
Dr. C. José Hernández Palancar

RT_031 **junio 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

REPORTE TÉCNICO
Reconocimiento de Patrones

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

Estado actual de los algoritmos de reconocimiento de rostro usando tecnología GPU

Ing. Danis López Naranjo,
Dr. C. José Hernández Palancar

RT_031 **junio 2010**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Índice

Índice de figuras

Índice de Tablas

Estado actual de los algoritmos de reconocimiento de rostro usando tecnología GPU

Ing. Danis López Naranjo, Dr. C. José Hernández Palancar

Centro de Aplicaciones de Tecnología de Avanzada,
7a #21812 e/ 218 y 222, Siboney, Playa, Habana, Cuba
dlopez@cenatav.co.cu

RT_031 CENATAV
Fecha del camera-ready: 26 de marzo de 2010

Resumen: En la actualidad se ha alcanzado un cierto nivel de madurez de los algoritmos y métodos que se han desarrollado en las aplicaciones dedicadas al reconocimiento de rostro, utilizando las unidades de procesamiento gráfico para acelerar estos algoritmos. Estas aplicaciones son capaces de identificar automáticamente a una persona mediante una imagen o un fotograma tomado de una secuencia de video por sus características faciales. Un ejemplo de estas aplicaciones en la actualidad lo podemos encontrar en: sistemas de vigilancia, proceso de autenticación de usuarios en aplicaciones informáticas, control de acceso, entre otras. Este estudio se centra en el estado actual de los algoritmos implementados en la GPU usados en el reconocimiento de rostros mostrando sus desventajas, ventajas y futuros temas a desarrollar.
Palabras clave: GPU, GPGPU, CUDA, pipeline

Abstract: At present, a certain level of maturity has been reached for the algorithms and methods that have been developed for applications dedicated to face recognition using graphics processing units to accelerate these algorithms. These applications are capable of automatically identify a person through an image or a frame taken from a video sequence by their facial features. An example of these applications today can be found in: surveillance systems, user authentication process in computer applications, access control, among others. This study focuses on the current state of algorithms implemented on the GPU used in face recognition showing its drawbacks, advantages and future issues to investigate.
Keywords: GPU, GPGPU,CUDA, Pipeline

1. Introducción

Actualmente, las unidades de procesamiento gráfico o GPU (Graphics Processing Unit) han alcanzado una notable evolución desde simples dispositivos de hardware para mostrar un gráfico en una pantalla, hasta convertirse en poderosas unidades de procesamiento gráfico con un alto sistema programable altamente paralelo con cientos de simples procesadores programables y una gran capacidad de memoria de ancho de banda. Con la necesidad de los desarrolladores de video juegos de imitar los entornos en tercera dimensión (3D) y realidad virtual en tiempo real y la imposibilidad de efectuar un cálculo óptimo en la unidad de procesamiento central o CPU (Central Processing Unit) de numerosas y complejas operaciones algebraicas, trajo como consecuencia la creación de nuevos dispositivos de hardware que sean capaces de resolver esta interrogante, como lo son las GPUs, que han revolucionado el mundo de los gráficos computarizados.

Las unidades de procesamiento gráfico cuentan con centenas de procesadores, esto le permite una gran capacidad de procesamiento explotado con éxito por desarrolladores y científicos en aplicaciones geométricas y de procesamiento de imágenes. Estas bondades como son el aumento significativo en el paralelismo dentro de un procesador no solo son utilizadas en el mundo de los gráficos, sino, en otras esferas como: la simulación física, la bioinformática, la minería de datos y el reconocimiento de patrones, siendo esta última el principal campo de acción donde se centra el estudio en particular el reconocimiento de rostro.

Esta investigación se enfoca en el análisis de la aceleración de algoritmos relacionados con el reconocimiento de rostros implementados, para ser ejecutados en las unidades de procesamiento gráfico utilizando los modelos de programación paralela. En épocas anteriores, muchos de estos algoritmos fueron programados en un lenguaje de bajo nivel como ensamblador; que por su difícil proceso de desarrollo fue necesario la reimplementación de varias librerías gráficas como: OpenGL (versión libre) y DirectX (de Microsoft) y lenguajes de programación de alto nivel tales como: Cg [3], HLSL [4], Sh [6], GLSL [5], SCOUT [8] diseñados específicamente para la visualización de datos en la GPU.

El estudio muestra algunos de los principales resultados y aspectos de interés relacionados con la aplicación de algoritmos acelerados e implementados en paralelo para el reconocimiento de rostros procesados por una GPU. Este trabajo se ha organizado el informe en las siguientes temáticas: 1ra sección: Unidad de Procesamiento Gráfico, 2da sección: Computación Paralela , 3ra sección: Está dedicada las unidades de procesamiento gráfico de propósito general, 4ta sección: Algoritmos de reconocimiento facial implementados en una GPU, 5ta sección: Se muestran los resultados obtenidos con esta investigación.

2. Unidad de procesamiento gráfico o GPU

Los productos de hardware gráficos han ido evolucionado considerablemente en las últimas dos décadas, por la incorporación de más capacidad de cálculo y programación a fin de atender las crecientes demandas de multimedia, video juegos, ingeniería y aplicaciones de visualización científica. Los primeros procesadores gráficos eran esencialmente dispositivos de función fija que implementaban un pipeline gráfico a la rasterización. Los dispositivos modernos son muy programables y ejecutan el software de forma muy similar a una CPU, cada generación de las GPU dependen cada vez más de grandes conjuntos de unidades de procesamiento completamente programables, en lugar del hardware con funciones fijas utilizadas en los dispositivos de la generación anterior.

En los últimos cinco años, se han alcanzado numerosos avances importantes en la adaptación de la arquitectura de hardware de la GPU para un mejor apoyo a la computación de propósito general, además de las cargas de trabajo existentes de gráficos. La creciente necesidad de programación de software para el procesamiento geométrico y otras operaciones más allá del sombreado de las GPU modernas, condujo a la transición de dejar de usar efectos especiales de formatos numéricos a las representaciones de la máquina estándar para los números enteros y números en punto flotante, junto con una simple y

doble precisión IEEE-754 del punto flotante a la capacidad de la aritmética comparable a la proporcionada por la instrucción de la CPU.

2.1. Arquitectura de la GPU

Desde la perspectiva física de estos dispositivos o hardware se puede afirmar que una unidad de procesamiento gráfico o GPU (Graphic Processing Units) se compone de un número limitado de multiprocesadores, cada uno de ellos consta de un conjunto de procesadores simples que operan de la forma Única Instrucción, Múltiples Datos o SIMD, es decir, todos los procesadores de un multiprocesador ejecutan de manera sincronizada la misma aritmética o la operación lógica al mismo tiempo, operando potencialmente sobre datos diferentes. Por ejemplo, la GPU de la generación más reciente GeForce GTX 295 con dos GPU y 30 multiprocesadores de 8 unidades de procesamiento cada uno, resumiendo un importe total de 480 procesadores dentro de una GPU. Su nivel de proceso alcanza casi los 2 teraflops, lo que la convierte en la tarjeta más rápida y potente diseñada hasta el momento. La GPU GTX 295 es la nueva generación de la arquitectura de NVIDIA Tesla, incorporado en la GeForce GTX serie 2x0. La arquitectura de la GPU GTX295 es similar en muchos aspectos a sus predecesores G80 y G92, que fueron los primeros con soporte para NVIDIA CUDA (Compute Unified Device Architecture) para el cálculo de la GPU. Al igual que con los diseños de la generación anterior, GTX 295 está compuesto de un gran número de unidades de procesamiento programable que se agrupan y comparten recursos. Los multiprocesadores en un grupo de procesador de textura se ejecutan completamente independiente el uno del otro, aunque comparten algunos recursos. La transmisión de los multiprocesadores de la GTX295 y sus predecesores aplican una sola instrucción de múltiples hilos de ejecución (SIMT) a una unidad de instrucción para lograr un uso eficiente.

Un rasgo clave de la arquitectura de las GPUs modernas es la utilización de múltiples sistemas de memoria de alto ancho de banda para mantener la gran variedad de unidades de procesamiento con los datos suministrados. La arquitectura GTX295 es compatible con un gran ancho de banda de (223.8 GB/s). El sistema de memoria principal es complementado con hardware dedicado a texturas que ofrecen las unidades de almacenamiento en caché de sólo lectura, con soporte de hardware para la localidad espacial multidimensional de referencia, de múltiples texturas de filtrado y modos de interpolación. La capacidad de 64 KB de caché constante es un medio eficaz de transmisión de sólo lectura de los elementos idénticos de datos a todos los temas dentro de un multiprocesador de transmisión a la velocidad de registro. El recuerdo constante puede ser una herramienta eficaz para lograr un alto rendimiento de los algoritmos que requieren recorrer y leer datos idénticos solamente. Como sus predecesores, la arquitectura GTX 295 incorpora un área de 16 KB de memoria compartida en cada uno de los multiprocesadores. Los subprocesos que se ejecutan en el mismo multiprocesador pueden cargar y manipular los bloques de datos en este registro usando la rápida velocidad de la memoria compartida, evitando costosos accesos a la memoria global. Los accesos a la zona de memoria compartida son coordinados a través del uso de una barrera de sincronización del hilo de ejecución primitivo, garantizando que todas las peticiones han terminado sus actualizaciones de memoria compartida antes de iniciar las discusiones de otros resultados para acceder a la misma. La memoria compartida

puede ser utilizado con eficacia como uno de los programas gestionados por la caché, la reducción de la utilización del ancho de banda y la latencia de las cargas repetitivas y tiendas para el sistema de memoria global.

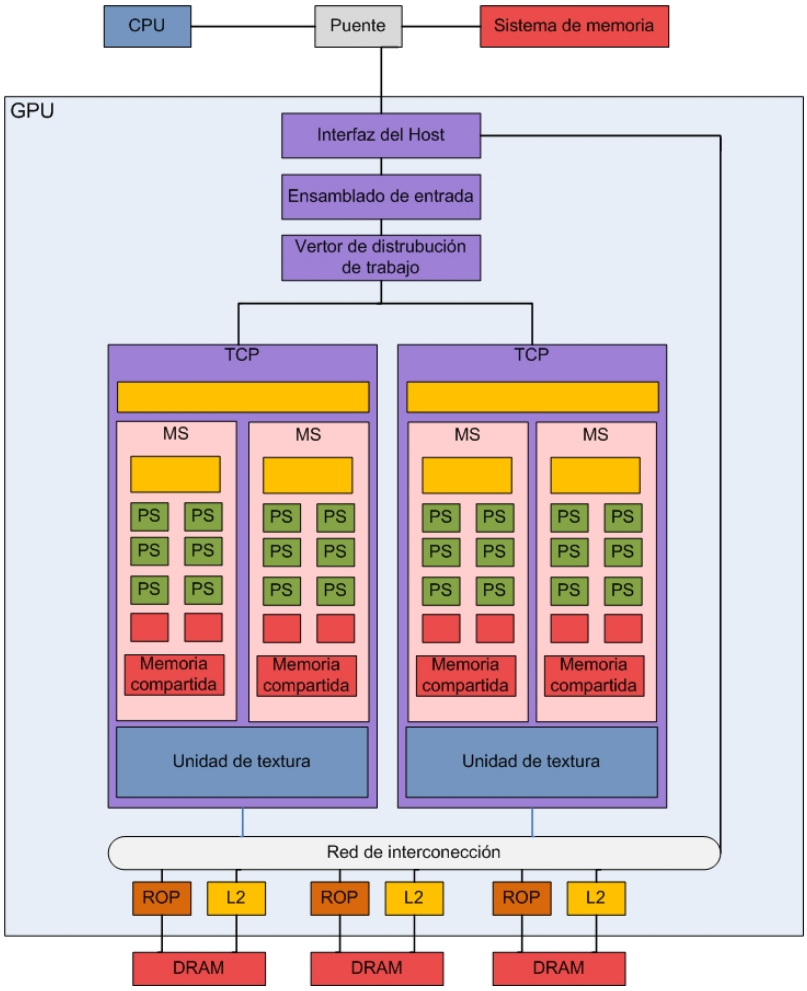


Fig. 1. Arquitectura de la GPU

2.2. Modelo de memoria

Aparte de algunas unidades de memoria con un propósito especial en el contexto de los gráficos, cuentan con tres grandes tipos de memoria, la memoria local (Local Memory), la memoria compartida (Shared Memory) que es una unidad de memoria de acceso rápido que se comparte entre todos los procesadores de un multiprocesador, esta no solo puede ser utilizada por las variables locales, sino también para el intercambio de información entre los hilos de ejecución en diferentes procesadores del mismo multiprocesador. Esta memoria no puede ser utilizada para la información que se comparte entre los diferentes hilos de ejecución en los multiprocesadores. La memoria compartida es muy rápida, pero tiene una

muy limitada capacidad (16 KB por multiprocesador). El tercer tipo de memoria es la memoria del dispositivo o DM (Device Memory), que es la memoria RAM de vídeo real de la tarjeta gráfica (se usa también para memoria de vídeo, etc.), esta es significativamente y mas lenta que la Memoria compartida. En particular, los accesos a la memoria causan un retraso de latencia típica de 400-600 ciclos de reloj (el G200-GPU, que corresponde a 300-500ns). El ancho de banda para transferir datos entre la memoria del dispositivo y GPU (223,8 GB/s en la GTX 295) es mayor que el de la CPU.

2.3. Pipeline gráfico

En la figura 2 se muestra la estructura simplificada del renderizado del pipeline gráfico de la GPU. Donde un procesador de vértices recibe un vértice de dato y luego es capaz de convertir una imagen vectorial en una foto computarizada que es construida usando los fragmentos a cada localización de pixel cubierto por una primitiva geométrica simple. Para lograr este objetivo el pipeline realiza las siguientes operaciones:

1. Procesar vértices.
2. Agrupar en primitivas simples.
3. Realizar las operaciones de, rotar, trasladar, escalar e iluminar.
4. Acotar texturas.
5. Mezclar elementos (blending).

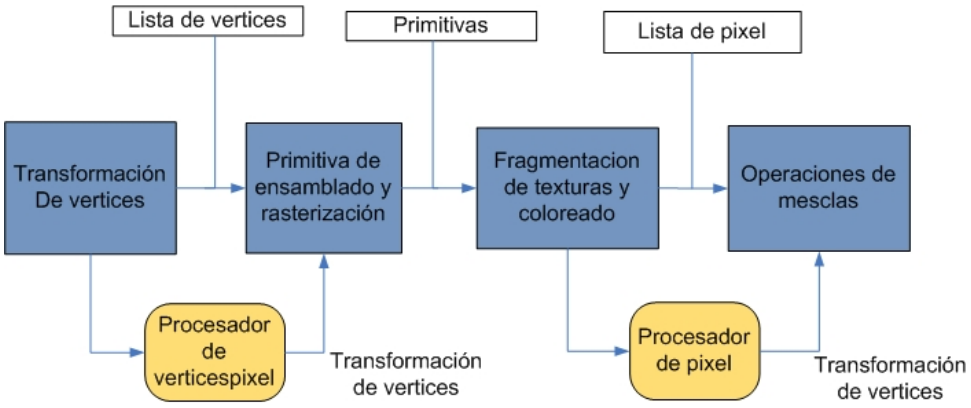


Fig. 2. Pipeline gráfico

2.4. Comparación con la CPU

Tradicionalmente la GPUs están diseñadas como unidades de procesamiento de rendimiento orientado al uso hardware sofisticado para programar la ejecución de decenas de miles de hilos simultáneamente en un gran número de unidades de procesamiento, o como su nombre lo indica una unidad de procesamiento gráfico de una computadora. Pero ¿Quién puede afirmar que las GPU solo se puede utilizar para la representación de gráficos? Muchas comunidades científicas han demostrado el uso extensivo de las GPU en otros campos de investigación con la simulación física, bioinformática y la Minería de Datos.

¿Por qué usar una GPU para realizar los cálculos generales?

La CPU está diseñada con memorias caché de gran tamaño para reducir la latencia hacia las localidades de memoria de acceso frecuente en la memoria principal. Estas memorias caché consumen una gran parte en los diseños de la CPU, espacio que podría utilizarse para las unidades de procesamiento adicionales de la aritmética. La GPU adopta un enfoque muy diferente y emplea el hardware multi-hilo de ejecución para ocultar la latencia de acceso a la memoria en el cambio de un subproceso ejecutable cuando el hilo de ejecución se encuentra con una activa dependencia de una operación a la espera de memoria global. Con un número suficiente de hilos multiplexados en cada una de las unidades de procesamiento, la latencia es realmente oculta, y el dispositivo logra un alto rendimiento sin la necesidad de una memoria caché. GPU utiliza el área que habrían sido consumidos por la memoria caché en la CPU para las unidades de la aritmética adicionales, dando un rendimiento de la arquitectura orientada a aritmética máxima de alto rendimiento.

El rendimiento de una GPU es mucho mayor que en la CPU. Una computadora personal o PC con una unidad central Pentium 4 a 3 GHz puede teóricamente publicar alrededor de 6 billones de puntos flotantes o float-point por segundos, sin embargo un fragmento de programa ejecutado en una GPU de modelo NVIDIA GeForce FX 5900 Ultra publica sobre los 20 billones de puntos flotantes por segundo, siendo esto lo equivalente a una CPU Pentium 4 a 10 GHz. En cuanto a la memoria de ancho de banda disponible, en la GPU puede alcanzar un tope de 25.3 GB/seg comparando con la Pentium 4 puede alcanzar los valores de 5.96 GB/seg. El balanceo de carga es una característica fundamental que hay que tener en cuenta, la CPU limita el rendimiento de la aplicación dejando la GPU con ciclos inactivos, limitando la descarga del trabajo a la GPU para lograr un aumento de velocidad en la aplicación. Por estas razones queda claro que la GPU es mucho mejor para realizar cálculos complejo que en una CPU. Una de las desventajas de la GPU es que no se pueden implementar todos los algoritmos en ella, ni pude ser vista como una plataforma de elección para la implementación de algoritmos, debido a su obligatoria implementación en paralelo. Sin embargo, el soporte de herramientas de programación para la GPU tales como: depuradores de cogido, lenguajes de programación son muy escasos que los que existen para la CPU.

2.5. Comparación con dispositivos similares: FPGA, Cell

Con el avance de las nuevas tecnologías de la computación, las tarjetas FPGA (Field Programmable Gate Array) han sido evolucionadas y mejoradas en la actualidad por sus desarrolladores, siendo cada vez más potentes, con una amplia velocidad de entrada y salida de datos con multiplicadores duros y abundantes bloques de memoria. El objetivo de usar estos dispositivos físicos para ordenadores, es proporcionar un modelo de programación que sea de fácil acceso para los programadores. El desarrollo de un diseño de un programa en un lenguaje de descripción de hardware, como Verilog es muy difícil, y requiere de un diseñador experto en hardware para realizar todas las implementaciones, pruebas, depuraciones de código necesarias para el desarrollo del hardware real. Las técnicas de síntesis que requiere un programador para escribir en un lenguaje de alto nivel como C, que luego se traducen automáticamente en los circuitos de hardware, en la actualidad tienen muchas limitaciones.

Las FPGA al igual que las GPU tienen varias diferencias técnicas de las cuales sobresalen, el cálculo del los puntos flotantes por la GPU, mientras que en las FPGA son más adecuadas para adaptar los puntos fijos o fixed-point.

Un artículo muy interesante donde establecen una comparación entre varios dispositivos de aceleración de algoritmos paralelos por hardware es [24], en el cual se evalúa el modelo SPICE en varias ocasiones para todos los dispositivos en el circuito. El compilador escogido usa estrategias de la arquitectura específica de paralelización como: OpenMP para multi-core, PThreads para Cell, CUDA para la GPU y VLIW para FPGA. Cuando el código que se producen para estas arquitecturas diferentes, los autores del artículo, automáticamente exploran las diferentes configuraciones para la aplicación usando su sintonizador para identificar la mejor configuración posible para cada arquitectura. Alcanzando valores de aceleraciones de 3-182x en un Xilinx Virtex5 LX 330T, 1.3-33x para un IBM Cell, y de un 3-131x en la GPU NVIDIA GeForce GT 9600 sobre en una CPU a 3 GHz Intel Xeon 5160 para una variedad de modelos de dispositivos de simple precisión. En [24] los autores muestran varios resultados del modelo SPICE implementado para varias arquitecturas, a continuación se muestran cuatro tablas de comparación de estos dispositivos en cuanto: al auto sincronización de parámetros, el máximo valor alcanzado de los puntos flotantes, el ambiente de software y el modelo de costo de los FPGA.

Tabla 1. Auto-Tuning Parameters publicada en [24]

Architecture	Parameter	Range (Step)
Intel	Loop-Unrool Factor	1-5 (1)
	MKL Vector	True/False
NVIDIA GPU	Loop-Unrool Factor	1-2 (1)
	Threads per block	1-2 (1)
IBM Cell	Loop-Unrool Factor	1-3 (1)
FPGA	Loop-Unrool Facto	1-15 (5)
	Operator per PE	8-64 (2*)
	BFT Rent Parameter	0.0-1.0 (0.1)

Tabla 2. Peak Floating-Point publicada en [24]

Family	Intel Xeon	Intel Core i7	Xilinx V6	IBM Cell	NVIDIA GPU	AMD GPU
Chip	5160	965	LX760	PS3	9600 GT	AMD 9270
Tecnology	65 nm	45 nm	40 nm	65 nm	65	55 nm
Clock	3 GHz	3.2 GHz	200 MHz	3.2 GHz	1.625 GHz	750 MHz
Double-Precision GFLOPS	12	25.6	26	10.5	-	240
Single-Precision GFLOPS	24	51.2	75.6	2004.8	312	1200
Power	80 Watts	130 Watts	20-30 Watts	135 Watts	59-96 Watts	160-220 Watts

Tabla 3. Software Enviroments publicada en [24]

Arch	Compiler	Libraries	Timing
Intel	Gcc-4.3 (-O3)	Libm, Intel MKL 10.1	PAPI 3.6.2 [26] PAPI_flops()
NVIDIA GPU	Nvcc CUDA SDK 2.1 [1]	CUDA libraries	cudaEventRecord()
IBM Cell	Spu-gcc ppu-gcc Cell SDK 3.1 [20]	Simdmath, MASS	Gettimeofday()
Xilinx FPGA	Synplify Pro 9.6.1 Xilinx ISE 10.1	CoreGen, Arenaire [2] [15]	-

Tabla 4. FPGA Cost Model publicada en [24]

	Area(Slices)	Latency(clocks)	Speed	Ref
Add	296	8	280	[2]
Multiply	611	9	237	[2]
Divide	1499	57	258	[2]
Square Root	822	57	282	[2]
Exponential	1022	30	200	[15]
Logarithm	1561	30	200	[15]
PE support logic	82	-	300	-
BFT T-Switchbox	48	2	300	-
BFT Pi-Switchbox	64	2	300	-
BFT Pi-Switchbox	64	2	300	-

Los autores de [24] son capaces de demostrar que los FPGAs superan todas las otras arquitecturas para dispositivos pequeños como, la altamente optimizada arquitectura personalizada VLIW que puede hacer un uso eficiente de los limitados recursos de las FPGAs. Para dispositivos más grandes, las GPU son capaces de proporcionar un rendimiento superior debido a menores gastos generales de programación y el mayor operador de paralelismo.

2.6. Modelo de programación

La principal característica de la GPU es que no es un procesador en serie, también conocido como arquitectura de von Neumann (von Neumann 1945), pero sí pertenecen al grupo de los procesadores de flujos o stram-processor, que se diferencian ligeramente por la ejecución de una función en un conjunto de registros de entrada produciendo un conjunto de archivos en paralelo. Estos procesadores se refieren a esta función como un módulo central de un sistema o kernel y al conjunto de registro como un flujo.

El modelo básico de programación de la GPUs es a través de los hilos de ejecución o threads en paralelo. Estos hilos de ejecución no son más que ligeros procesos que son fáciles de crear y sincronizar a diferencia de los procesos de la Unidad de Procesamiento Central o CPU (Central Processing Unit).

La biblioteca de programación CUDA contiene funciones de la API para crear un gran número de hilos en la GPU, cada uno de los cuales ejecuta una función llamada función de núcleo (kernel function). Las funciones del núcleo que se ejecutan en paralelo en la GPU se definen en una sintaxis extendida del lenguaje de programación C. Las funciones del núcleo están restringidas con respecto a la funcionalidad (por ejemplo, sin recursión).

2.6.1. CUDA

Los programas desarrollados para ser ejecutados por la GPU, eran implementados en lenguajes de bajo nivel o se necesitaba un amplio conocimiento de programación avanzada y bibliotecas de librerías gráficas para implementarlos en lenguajes de alto nivel. Por lo tedioso y complicado que resultaba el desarrollo de estos programas, los desarrolladores de NVIDIA crearon una nueva tecnología denominada NVIDIA CUDA (Compute Unified Device Architecture), siendo esta el único entorno de programación en C que aprovecha la gran capacidad de procesamiento de las GPU para resolver los diversos problemas de cálculos complejos y de mayor carga computacional, sin necesidad de mapear a una API

de cualquier librería gráfica. Esta tecnología está disponible desde la serie 8 de GeForce hasta la GPU más moderna de la empresa NVIDIA. El mecanismo multitarea del sistema operativo es el responsable de mapear el acceso a varias tecnologías CUDA y aplicaciones gráficas ejecutadas concurrentemente.

Cuando se programa la unidad de procesamiento gráfico con la Arquitectura Unificada de Dispositivos de Cómputo o CUDA, la GPU es vista como un dispositivo capaz de ejecutar un alto número de hilos de ejecución en paralelo y operar como un coprocesador de la CPU. En la ejecución de un programa, una porción de él es ejecutado en muchos tiempos en diferentes datos, este puede ser dispersado dentro de una función o kernel que está siendo ejecutada en el dispositivo como muchos hilos de ejecución diferentes. Tanto la memoria del ordenador (host memory) como la memoria del dispositivo (device memory) pueden copiar datos de la memoria de acceso aleatorio dinámica o DRAM (Dynamic Random Access Memory) entre ellas, utilizando las llamadas optimizadas de la API del dispositivo de alto-rendimiento (high-performance) de la memoria de acceso directo o DMA (Direct Memory Access).

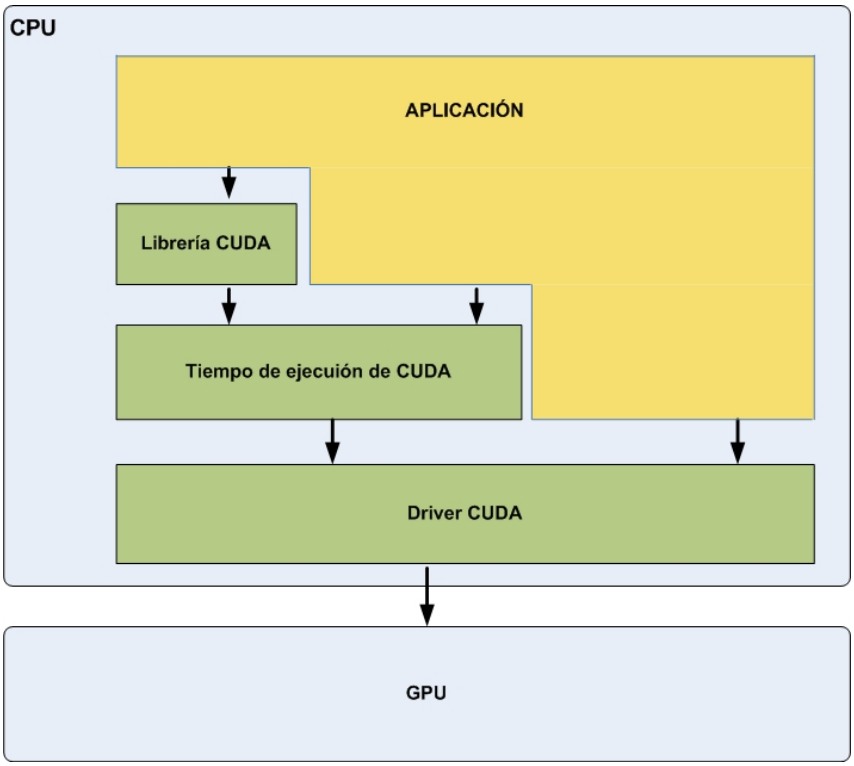


Fig. 3. Arquitectura CUDA

2.6.1.1. Modelo de memoria

Un hilo de ejecución que se ejecuta en el dispositivo sólo tiene acceso al dispositivo DRAM y al on-chip de memoria que pasa por los siguientes espacios:

Tabla 5. Acceso a memoria usando CUDA

Permiso	Grupo	Acceso
Lectura y escritura	Hilos de ejecución	Registros
Lectura y escritura	Hilos de ejecución	Memoria local
Lectura y escritura	Bloques de hilos	Memoria compartida
Lectura y escritura	Redes de Bloques	Memoria Global
Lectura	Redes de Bloques	Memoria constante
Lectura	Redes de Bloques	Memoria de textura

2.6.1.2. Implementación del dispositivo físico

Estos dispositivos son implementados respetando el paradigma de programación paralela *Única Instrucción, Múltiples Datos* o SIMD (Simgle Instruction, Multiple Data) como lo muestra la siguiente figura. Cada multiprocesador cuenta con un chip de memoria de los siguientes tipos:

1. Un conjunto de registros locales de 32-bit por procesadores.
2. La memoria caché de los datos paralelos o memoria compartida que está disponible por todos los procesos e implementan los espacios de memoria compartida.
3. Una memoria caché constante de solo lectura que es compartida por todos los procesadores y aceleradores de lectura del espacio de la memoria constante, que se implementa como una lectura única de la región del dispositivo de memoria.

2.6.1.3. Capacidad de cómputo

La capacidad informática de un dispositivo se define por un número de revisión de mayor y menor importancia. Los dispositivos con el mismo número de revisiones más importantes son de la misma arquitectura de núcleo. Los productos enumerados en la tabla 6 son todos de la capacidad de calcular 1.x (de su mayor número de revisión es 1). El número menor de la revisión corresponde a una mejora progresiva de la arquitectura del núcleo, que pueden incluir nuevas características.

2.6.1.4. Operaciones atómicas

Con el fin de sincronizar todos los procesos paralelos y asegurar la exactitud de algoritmos paralelos, CUDA ofrece operaciones atómicas tales como: incremento, decremento, o el intercambio entre otras operaciones. Desde la versión de CUDA 1.3 hasta la versión actual permiten las operaciones atómicas en un multiprocesador o SM (Streamer Multiprocessor), por ejemplo, algunos procesos paralelos comparten una lista como un recurso común con concurrencias como leer y escribir desde la lista, puede ser necesariamente atómico el incremento de un contador para el número de entradas de la lista. La atomicidad implica, en este caso, los dos requisitos siguientes: Si dos o más hilos incrementan el contador de la lista, a continuación, [1] lucha contra el valor después de todos los incrementos concurrentes debe ser equivalente a la del valor de antes más el número de operaciones de incremento concurrente. Y en [21] cada uno de los hilos concurrentes debe obtener un

Tabla 6. Capacidad de cómputo para las GPUs NVIDIA publicado en [1]

GPU NVIDIA	No de multiprocesadores	Capasidad de cómputo
GeForce 8800 Ultra, 8800 GTX	16	1.0
GeForce 8800 GT	14	1.1
GeForce 8800 MGTX	12	1.1
GeForce 8800 GTS	12	1.0
GeForce 8800 MGTS	8	1.1
GeForce 8800 MGTX	4	1.1
GeForce 8600 GT, 8600GS 8700MGT, 8600MGS	2	1.1
GeForce 8400 MG	2	1.1
Tesla S870	4x16	1.0
Tesla D870	2x16	1.0
Tesla C870	16	1.0
Quadro Plex 1000 Model S4	4x16	1.0
Quadro Plex 1000 Model IV	2x16	1.0
Quadro FX 5600	16	1.0
Quadro FX 4600	12	1.0
Quadro FX 1700, Fx 570 NVS 320,FX 1600M, FX570M	4	1.1
Quadro FX 370,NVS 140M NVS 290, FX 360M, NVS 135	16	1.0
Quadro NVS 130M	1	1.1
GeForce 8800 Ultra, 8800 GTX	16	1.0
GeForce 8800 GT	14	1.1
GeForce 8800 MGTX	12	1.1
GeForce 8800 GTS	12	1.0

resultado independiente del incremento de operación, que indica el índice del elemento de la lista vacía para que el hilo pueda escribir su información. Por lo tanto, la mayoría de operaciones atómicas devuelven un resultado después de su ejecución. Los dispositivos de capacidad de cómputo 1.1 como lo muestra la anterior tabla 7 son los que están diseñados para ejecutar operaciones atómicas.

2.7. Paralelismo en la GPU

Muchas aplicaciones de grandes conjuntos de datos como los arreglos de datos, pueden utilizar un modelo de programación paralela para acelerar los cálculos. En la representación 3D de grandes conjuntos de píxeles y vértices se asignan a las discusiones paralelas. Del mismo modo, la imagen y los medios de comunicación como la tramitación de solicitudes de post-procesamiento de las imágenes renderizadas, la codificación y descodificación de vídeo, escalado de imagen, la visión estéreo, y el reconocimiento de patrones pueden asignar bloques de la imagen y los píxeles de hilos de procesamiento en paralelo. De hecho, muchos algoritmos fuera del ámbito de la representación y el procesamiento de imágenes pueden acelerados por el procesamiento de datos en paralelo de las unidades de procesamiento gráfico como se muestra más adelante en el documento, como son: el procesamiento de señales en general o de simulación física para la financiación de cálculo o biología computacional.

Las aplicaciones de rendimiento tienen varias propiedades que las distinguen de las aplica-

ciones de serie de la CPU tales como:

1. Abundantes datos sobre miles de cálculos paralelo sobre los elementos de datos independientes.

2. Los grupos de paralelismo de los hilos de ejecución ejecutan el mismo programa, y los diferentes grupos pueden correr diferentes programas.

3. Intensiva aritmética del punto flotante.

4. La tolerancia de latencia del rendimiento es la cantidad de trabajo realizado en un momento determinado.

5. La transmisión del flujo de dato requiere una alta memoria de ancho de banda con poca reutilización de datos relativamente.

6. La modesta sincronización entre los hilos de ejecución y la comunicación gráfica no se comunican, y las aplicaciones de computación paralela requieren de una limitada sincronización y comunicación.

Las unidades de procesamiento gráfico proporcionan un tipo de restricción del procesamiento en paralelo referido al modelo de Instrucción Única Múltiples Datos o SIMD (Single Instruction, Multiple Data) o siendo un poco más específicos se le puede llamar a este modelo Único Programa, Múltiples Datos o SPMD (Single Program, Multiple Data). Cada uno de muchos procesos son ejecutados simultáneamente de algún programa en diferentes elementos de datos.

3. Computación paralela

El paralelismo en aplicaciones informáticas a veces se ve con poca relevancia para el programador medio. Un estudio de las tendencias en las aplicaciones informáticas muestra que esta visión ya no es sostenible. El paralelismo se está expandiendo en las últimas décadas, y la programación en paralelo se está convirtiendo en una buena opción para las optimizaciones de las aplicaciones. De manera informal se puede definir, que una computadora en paralelo es un conjunto de procesadores que son capaces de trabajar colaborativamente para resolver un problema computacional. Tradicionalmente, en la ciencia de la informática las aplicaciones paralelas han sido motivadas por las simulaciones numéricas de sistemas complejos, como los dispositivos mecánicos, circuitos electrónicos, procesos de fabricación, las reacciones químicas. En la actualidad se destacan las aplicaciones comerciales como los principales impulsadores de esta rama que requieren de una computadora para ser capaces de procesar grandes cantidades de datos en formas sofisticadas. Dentro de estas aplicaciones podemos encontrar la video conferencia, entornos de trabajo colaborativo, bases de datos paralelas utilizadas para apoyar las decisiones, gráficos avanzados y la realidad virtual, en particular en la industria del entretenimiento.

De lo expuesto anteriormente se puede afirmar, que el paralelismo no está de manifiesto solamente en las supercomputadoras, sino también, en las estaciones de trabajo, computadoras personales y redes. Ya en la actualidad los programas son necesarios para explotar los múltiples procesadores ubicados dentro de cada equipo. Debido a que la mayoría de los algoritmos existentes están especializados en un único procesador, esta situación implica la necesidad de nuevos algoritmos y estructuras de programas capaces de realizar muchas

operaciones a la vez. Concurrencia que se convierte en un requisito fundamental para los nuevos algoritmos y programas [16].

A medida que pasa el tiempo los procesadores seguirán mejorando su capacidad de procesamiento y su cantidad por equipo. Ian Foster plantea que la escalabilidad es tan importante como la portabilidad para la protección de las inversiones en software y que un programa capaz de utilizar sólo un número fijo de procesadores es un mal programa, como lo es un programa capaz de ejecutarse en una sola computadora [16].

La rápida penetración de los ordenadores en el comercio, la ciencia y la educación debe mucho a la normalización de principios en un modelo de máquina, la computadora de Von Neuman. Una computadora de Von Neuman incluye una unidad de procesamiento central (CPU) conectada a una unidad de almacenamiento. La CPU ejecuta un programa almacenado que específica una secuencia de lectura y escritura en la memoria. Este modelo simple ha demostrado ser extremadamente robusto. Su persistencia durante más de cincuenta años ha permitido el estudio de temas tan importantes como los algoritmos y los lenguajes de programación para proceder, en gran medida, independientemente de la evolución de la arquitectura de computadores. En consecuencia, los programadores pueden ser entrenados en el arte abstracto de la programación, más que el arte de la programación de la máquina " X" y pueden diseñar algoritmos para una máquina abstracta de Von Neuman, confiando en que estos algoritmos se ejecutarán en la mayoría de ordenadores de destino con una eficiencia razonable.

La programación en paralelo introduce fuentes adicionales de complejidad por ejemplo: si tuviéramos que programar a bajo nivel, no sólo aumentaría el número de instrucciones ejecutadas, sino también tendría que gestionar de manera explícita de la ejecución de miles de procesadores y coordinar a millones de interacciones entre los procesadores. Por lo tanto, la abstracción y la modularidad son al menos tan importantes como en la programación secuencial[16].

3.1. Velocidad

Para lograr una mejora en la velocidad mediante el uso de paralelismo, es necesario dividir el cálculo en las tareas o procesos que pueden ser ejecutadas simultáneamente. Se pude afirmar, que un programa es escalable si mantiene la propiedad que reduce su tiempo de ejecución a razón de que que aumente su cantidad de procesadores, debido a que la velocidad es inversamente proporcional a la cantidad de procesadores. El tamaño de un proceso puede ser descrito por su granularidad. En la granularidad gruesa, cada proceso contiene un gran número de instrucciones secuenciales y toma un tiempo considerable a ejecutar. En la granularidad, un proceso puede consistir en unas pocas instrucciones, o tal vez incluso una instrucción. En algún momento la granularidad se define como el tamaño de la computación entre la comunicación o los puntos de sincronización. Una medida de rendimiento relativa entre un sistema multiprocesador y el sistema de un único procesador es el factor del aumento de velocidad, definida como:

$$S_{(n)} = \frac{T_s}{T_p}$$

Donde T_s es el tiempo de ejecución de un procesador y T_p es el tiempo de un multiprocesador con n procesadores. La expresión $S_{(n)}$ da el incremento en la velocidad de la utilización de un multiprocesador.

3.2. Aceleración máxima

La aceleración máxima con n procesadores es de n, también es conosida como **aceleración lineal**. Donde, si un algoritmo paralelo logró un mejor rendimiento de n veces la aceleración del algoritmo secuencial actual, entonces, el algoritmo paralelo sin duda puede ser emulado en un único procesador, lo que sugiere, que el algoritmo secuencial original no era óptimo. La aceleración máxima absoluta de n se logrará cuando el cálculo se pueda dividir en procesos de igual duración, con un proceso de mapeado en un solo procesador que está dado por la fórmula:

$S_{(n)} = \frac{T_s}{T_s/n} = n$

3.3. Eficiencia

Los sistemas eficientes se pueden definir por el tiempo de ejecución de un procesador (t_s) dividido en el tiempo de ejecución del multiprocesador (t_p) multiplicado por la cantidad de procesadores (n) que contenga como se muestra en la siguiente función:

$$E = \frac{t_s}{t_p * n}$$

que conduce a: $E = \frac{t_s}{t_p} * 100\,\%$

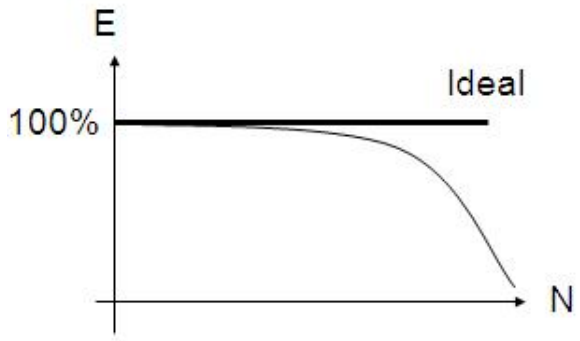


Fig. 4. Gráfica de la Eficiencia

3.4. Arquitectura

En el mundo de la computación paralela entre las principales arquitecturas que sobresalen en la clasificación de los sistemas paralelos se destacan:

1. Arreglos de Procesadores, también conocidas como máquinas MPP (massively par-

allel procesing, más de 1000 CPU).

2. Multiprocesadores (SMP) o máquinas de memoria compartida.

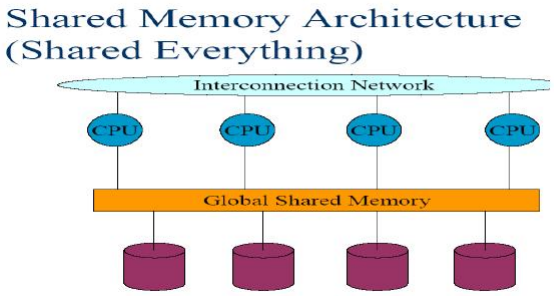
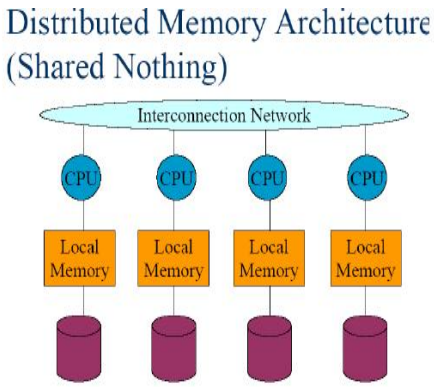
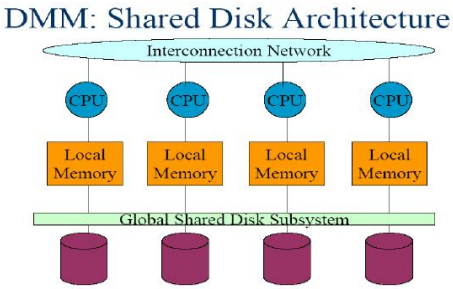


Fig. 5. Multiprocesadores (SMP) o máquinas de memoria compartida

3. Multicomputadoras memoria distribuida (cantidad de CPU menor o igual a 1000), también se les conoce como máquinas de memoria distribuida (distributed memory machines DMM).



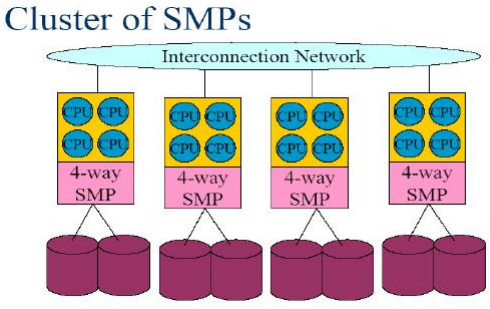
(a) Shared Nothing



(b) Shared Disk

Fig. 6. Arquitectura memoria compartida

4. Clusters de SMPs



4. Unidad de procesamiento gráfico de propósito general o GPGPU

Teóricamente, las GPUs son capaces de realizar cualquier cálculo que puede transformar el modelo de paralelismo y que permitan la arquitectura específica de la GPU. Este modelo ha sido explotado por múltiples áreas de investigación. En [35] presentan un nuevo enfoque para las simulaciones dinámica de alto rendimiento en unidades de procesamiento gráfico mediante el uso de CUDA en el diseño molecular y aplicar un nuevo algoritmo paralelo. Sus resultados indican una mejora de rendimiento significativo en una tarjeta NVIDIA GeForce GTX 8800 en el procesamiento secuencial en la CPU.

Otro documento interesante sobre los cálculos del campo de las ciencias de la vida ha sido publicado por Manavski y Valle en [30]. Los autores proponen una solución muy rápida del algoritmo Smith-Waterman, de un procedimiento que resuelve la búsqueda de similitudes en proteínas y bases de datos de ADN. Este algoritmo se ejecuta en la GPU y aplica en el entorno de programación CUDA, alcanzando aceleraciones importantes en una estación de trabajo con dos GPUs del modelo GeForce GTX 8800.

Otra área de aplicación generalizada que utiliza la potencia del alto procesamiento de la GPU es la simulación mecánica. Un ejemplo es el trabajo de Tascora en [7], que presenta un método novedoso para la solución de grandes problemas de la complementariedad de cono por medio de un algoritmo de punto fijo de la iteración, en el contexto de la simulación de la dinámica de rozamiento de los grandes sistemas de cuerpos rígidos. Para demostrar las posibilidades casi ilimitadas de realizar cálculos en la GPU, presentamos un ejemplo más, la informática criptográficas como se puede apreciar en [21]. En este documento, los autores presentan un rendimiento récord para la minería de datos con el uso de 67 unidades de procesamiento gráfico en la ejecución del método de curvas elípticas (ECM) de la factorización de números enteros. La aceleración se beneficia de dos GPU modelo NVIDIA GTX 295 actualmente la unidad de procesamiento más rápida de la historia, utilizando una aplicación de ECM nuevo basándose en nuevas fórmulas de adición en paralelo y las funciones que se encuentran disponibles por CUDA.

Algunos documentos proponen técnicas para acelerar las operaciones de base de datos relacional en la GPU. En [9] explican algunos algoritmos para la unión de tablas en bases de datos relacionales, usando una GPU NVIDIA G80 con el modelo de programación CUDA. Dos documentos recientes [13] y [27] abordan el tema de unirse a la similitud en el espacio que determina la característica de todos los pares de objetos de dos conjuntos diferentes, R y S que cumplen con una unión de predicado o (predicate-join). Lo más común es que la unión de predicado determina todos los pares de objetos que tienen una distancia de menos de un umbral predefinido. Los autores de [13] proponen un algoritmo basado en el concepto de espacio de llenado curvas, por ejemplo, el orden z, para la poda del espacio de búsqueda, trabajando en una GeForce 8800 de NVIDIA utiliza el kit de herramientas CUDA. El orden z de un conjunto de objetos puede ser determinado de manera muy eficiente en la GPU de alta paralelizada clasificación. Su algoritmo opera sobre una serie de Z-listas de granularidad de poda eficiente. Sin embargo, desde todas las dimensiones son tratados por igual, el rendimiento se degrada en dimensiones mayores. Además, debido a la compartimentación del espacio uniforme en todos los ámbitos del espacio de datos, el espacio de las curvas de llenado no son adecuadas para los datos agrupados.

Un enfoque que supere ese tipo de problema se presenta en [27]. Aquí los autores paralelizan la técnica base que subyace en cualquier operación de combinación con un predicado arbitrario, es decir, la combinación de ciclo anidado (NLJ), es una potente base de datos de primitivos que pueden ser utilizados para apoyar a muchas aplicaciones, incluyendo la minería de datos. Todos los experimentos se realizan en los procesadores gráficos NVIDIA GT 8500 por el uso de una aplicación compatible con CUDA. Govindaraju [18] demuestran que elementos importantes para el procesamiento de consultas en bases de datos, por ejemplo, clasificación, las selecciones conjuntivas, agregaciones, consultas semi-lineal se puede acelerar significativamente por el uso de las GPUs.

En [36] el principal objetivo es desarrollar un diseño genérico de la GPGPU que pueda ser utilizados para la aplicación de métodos de detección de muchos objetos que comparten una similar estructura de la ventana de exploración. En el artículo se centran en la aplicación de ese diseño para la detección de los peatones, sin embargo, se puede aplicar a cualquier método de detección de objetos que utiliza el paradigma de la exploración de la ventana.

El algoritmo presentado por Dalal en [14] se considera como una línea base estable, ya que consta con buenos resultados y sencillez de su método utiliza características HOG (histograma de los gradientes) y un clasificador SVM (Support Vector Machine). Muchos de los nuevos tipos de características, [10], [22], [28], y clasificadores han sido juzgados en el marco de la ventana de exploración para lograr un mejor rendimiento de la detección.

La agrupación es una de las principales aéreas de la minería de datos que usan la GPU para acelerar sus algoritmos y alcanzar un mejor rendimiento. CUDA fue el modelo de programación escogido por Cao F. en [12] donde plantea un enfoque de agrupación en una unidad de procesamiento gráfico del modelo NVIDIA GeForce GT 6800, que extiende la idea básica del algoritmo K-means para calcular las distancias de una simple entrada del baricentro a todos los objetos, a la vez que se puede hacer de forma simultánea en la GPU. Así, los autores son capaces de aprovechar la potencia de cálculo y el pipeline de la GPU, especialmente para las operaciones básicas, como los cálculos de distancia y las comparaciones. El artículo [31] se paraleliza el algoritmo K-means para el uso de una GPU mediante el uso de múltiples pasos de varios programas. La aplicación fue acelerada en los procesadores gráficos NVIDIA 5900 y NVIDIA 8500 logrando importantes resultados cada vez mayor para los dos procesadores con diferentes tamaños de datos y tamaños de clúster. Sin embargo, los algoritmos de ambos documentos no son transferibles a distintos modelos de GPU.

Según NVIDIA, que es la empresa líder en el mercado en esta tecnología, publicó en el año 2008 un conjunto de aplicaciones portadas a GPU como se muestran en la tabla 7:

Tabla 7. Aplicaciones de la GPU en la actualidad [24]

3D Image Analysis	Broadcast	Adaptive radiation therapy	Radar
Acoustics	Astronomy	Audio	Wireless
Automobile visionI	Bioinformatics	Biological simulation	Video
Cellular automata	Computational Fluid	Dynamics	Satellite data analysis
Computer Vision	Cryptography	CT Reconstruction	Reservoir simulation
Data Mining	Film	Digital cinema/projections	Quantum chemistry
Financial	Equity training	Electromagnetic interference	Oceanographic research
Holographics cinema	Languages	GIS	Network processing

Tabla 8. Más aplicaciones de la GPU

Military	Imaging	Mathematics research	Ray tracing
Mine planning	Optical inspection	Molecular dynamics	Ultrasound
Nbody	Particle physics	Multispectral imaging	MRI reconstruction
Neural networks	Surgery simulation	Protein folding	Video conferencing
Robotic vision/AI	Seismic imaging	Robotic surgery	Visualization
Surveillance	Telescope	X-ray	

5. Reconocimiento de rostros

En la década del 2000 se ha alcanzado un cierto nivel de madurez de los algoritmos y métodos que se han desarrollado en el reconocimiento de rostro utilizando a las unidades de procesamiento gráfico para acelerar estos algoritmos, las aplicaciones de este tipo están dadas básicamente por un creciente interés comercial por su gran utilidad en el mundo automatizado; una aplicación informática dedicada al reconocimiento de rostros humanos es capaz de identificar automáticamente a una persona mediante una imagen o un fotograma tomado de una secuencia de video por sus características faciales. Un ejemplo de estas aplicaciones en la actualidad lo podemos encontrar en: sistemas de vigilancia, proceso de autenticación de usuarios en aplicaciones informáticas, control de acceso, entre otras.

Los sistemas actuales se limitan típicamente a una tarea muy específica y de un ambiente controlado, la falta de generalidad es necesitada por las aplicaciones del mundo real. Esto incluye sobre todo la robustez frente a variaciones en el tamaño, posición, postura y la iluminación de la apariencia de la cara en la imágen resultante. Si bien la apariencia y función de muchos enfoques han sido desarrolladas y son capaces de resolver algunos de estos temas, la mayoría de ellos suelen fallar cuando se producen la variación de la iluminación y la postura. Una tendencia reciente a superar esas limitaciones, es explotar la información 3D de rostros humanos, que se puede obtener directamente del rango de un escanner [23], desde una imagen [11] o de múltiples imágenes [17]. Pero el uso de estas técnicas requiere un procedimiento iterativo no lineal que lleva mucho tiempo y no se ajusta a las características de las aplicaciones en el mundo real. Los antiguos métodos generalmente no son capaces de tratar con extrema variación de pose o no son aplicables a los escenarios del mundo real debido a limitaciones de tiempo, tales como procedimientos de optimización de consumo o complejos sistemas de adquisición de profundidad. En este capítulo se muestra el estado actual de los algoritmos utilizados en el reconocimiento de rostros acelerados en las unidades de procesamiento gráfico.

5.1. Taxonomía

5.2. Algoritmos dependientes de la pose

5.2.1. Basados en características geométricas

Este artículo [32] describe las implementaciones de la novedosa función de seguimiento *KLT* y los algoritmos de extracción característica *SIFT* que se ejecutan en la unidad de procesamiento gráfico y son adecuados para el análisis de vídeo en tiempo real en sistemas de visión. Estos algoritmos funcionan para las arquitectura de las GPUs ATI

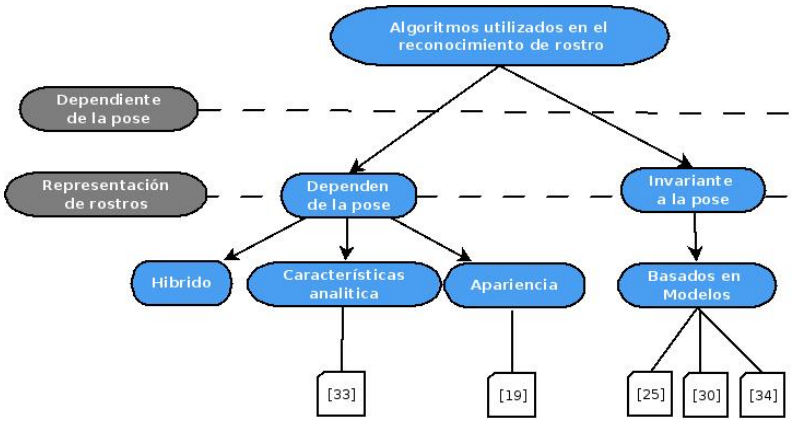


Fig. 7. Taxonomía de los algoritmos usados en el reconocimiento de rostro implementado en la GPU.

y Nvidia. De manera que, la extracción saliente de los puntos característicos 2D en el video son importantes en muchas fases de la visión por computadora, dentro de ellas resaltan la detención, el reconocimiento y la estructura del movimiento. El algoritmo de seguimiento KLT calcula el desplazamiento de las características de los puntos de interés entre los fotogramas de videos consecutivos, cuando la imagen de la restricción cumple la constancia de brillo y el movimiento en la imagen es bastante pequeño. El ejemplo citado por los autores se basa en un modelo de traslación local entre los fotogramas de videos consecutivos. El desplazamiento de una característica se calcula utilizando el método de Newton para minimizar la suma de las distancias al cuadrado (SSD) en una ventana de seguimiento alrededor de la posición característica de las dos imágenes.

Un fotograma de video en el tiempo está representado por la siguiente expresión $I(*,*,t)$. Si el desplazamiento de un punto (x,y) de la imagen es pequeño y está entre (t y $t+\triangle t$) denotado por $(\triangle x, \triangle y)$ y concuerda con la constancia de brillo se puede afirmar que:

$$I(x,y,t+\triangle t) = I(x+\triangle x, y+\triangle y, t)$$

Para $\mathbf{x} = (x,y)^t$ y $\mathbf{v} = (\triangle x, \triangle y)^t$ en presencia del ruido de la imagen (r) tenemos que:

$$I(\mathbf{x}, t+\triangle t) = I(\mathbf{x}+\mathbf{d}, t) + r$$

El algoritmo KLT calculará el vector de desplazamiento $\mathbf{d}$ que minimiza el siguiente error:

$$r = \sum_w (I(\mathbf{x}+\mathbf{d}, t) - I(\mathbf{x}, t+\triangle t))^2$$

Sobre una pequeña trayectoria de la imagen W. De la aproximación de $I(x+d,t)$ desarrollado por Taylor se obtiene el siguiente sistema de ecuaciones lineales para estimar d.

$\sum_w (G^t G)(d) = \sum_w (G^t \triangle I)$

Un poco más tarde Tomasi propuso una variación de la ecuación del KLT que utiliza ambas imágenes de forma simétrica. Para un mayor análisis de este algoritmo se muestran dos fragmentos del pseudo-código del mismo.

1. KLT Tracking

   KLTTracking(ft_list, F_0, F_1){
   (1) Build-Pyramid: builds multi-resolution intesity and gradient pyramid from imageF_0, F_1
   (2) Track:
   For all pyramid levels from coarse to fine
   For wach feature f in ft_list\{
   compute coefficients of A and b
   solve $Ad = b$
   evaluate d and update Track of feature
   \}
   }

2. Re-select-Feature
   Re-select-Feature (ft_list){
   $mask = mask\_out\_region(ft\_list)$
   c_map = evaluate cornes-ness measure c over whoile image
   //Perform non-maximal suppression
   $pts = find\_feature(\#max\_feats, mask, sort(c\_map))$
   add_new_feature(ft_list, pts)
   \}

La implementación de algoritmo KLT implementado para GPU mapea varias etapas del algoritmo de seguimiento a diferentes fragmentos de programas. Donde cada fragmento de video es cargado en una pirámide de multiresolución de la intensidad de la imagen y del gradiente. La construcción de esta pirámide es calculada por una serie de dos puntos separables por una convolución Gaussina ejecutadas en un fragmento del programa. El segundo conjunto de texturas se utilizan para almacenar los resultados de la convolución de la fila y pasar posteriormente a leer la convolución de la columna pasada.

El algoritmo SIFT-GPU de extracción de características es muy usado para la extracción de los puntos invariables a la translación, rotación, escalado y los cambios de iluminación de la imagen. Para lograr su implementación se construye una pirámide Gaussiana escalado-espacio de la imagen de entrada mientras se calculan los gradientes y la diferencia gaussiana o DOG (difference-of-gaussian) de las imágenes en estas escalas. Los puntos de interés se detectan en los extremos locales en el espacio de la escala DOG. Una vez que varios puntos claves se hayan detectado en diferentes escalas, los gradientes de imagen en la región local alrededor de cada punto de la función están codificados utilizando los histogramas de orientación y representación en forma de un descriptor de elementos invariante rotacional.

5.2.2. Basados en apariencia

El algoritmo mostrado [19] sintetiza la imagen de un rostro utilizando un modelo 3D morphable y minimiza el error entre la imagen de síntesis y la imagen de entrada al sistema extraída de un fotograma de video. La esencia de la demostración es, que se introduce una coordenada de minimización de error orientada mediante la adaptación de una solución lineal para estimar los parámetros de seguimiento. Con el fin de medir el error y minimizarlo, se muestra una técnica de optimización de multa. Además, se demuestra que el marco de trabajo propuesto reduce las complicaciones en el tratamiento de la variabilidad de la iluminación sobre la superficie del objeto que puede ser causada por ejemplo por el movimiento del objeto en sí. El sistema fue implementado en un ordenador del modelo macintosh con 2 CPUs (Power PC a 1,25 GHz) y una GPU GeForce4. La imagen es capturada por la aplicación usando una cámara Sony DV-VX2000 y después es recortada de una resolución de 640 x 480 a 512 x 480 en escalas de gris. El sistema es capaz de ejecutar un procedimiento de ajuste y mostrar un resultado 15 veces por segundo y de realizar un seguimiento en tiempo real de un rostro humano no rígido con gran precisión, gracias a los avances clave en:

1. El marco de trabajo utilizado no es más que la combinación del análisis-por-síntesis de enfoque que usa un modelo 3D morphable, que consiste en un número pequeño pero suficiente de las bases (morph) para expresar la estructura facial individual, y el uso de la minimización del error del método lineal.

2. La reducción del tiempo de cálculo para la técnica gruesa-fina y la computación paralela.

3. El diseño modular que permite diferentes técnicas de estimación del medio ambiente de iluminación con el fin de hacer frente a la variabilidad de la iluminación sobre la superficie de destino.

De este artículo se puede resaltar que queda para futuros trabajos, la demostración de la robustez del procesamiento de ajustes mostrado contra la oclusión parcial y/o la variedad de la iluminación.

5.3. Algoritmos que no dependen de la pose

5.3.1. Basados en modelos

El modelado de la cara humana y la piel es un tema ampliamente estudiado. Una visión general de las cuestiones relacionadas con el modelado de la piel está dada por Igarashi en "The appearance of human skin" publicado en el año 2005. Los enfoques existentes en él se pueden dividir en dos categorías principales: los basados en datos y en modelos. Los guiados por datos requieren información adicional de un escáner facial en 3D que sirven para medir la función de reflectancia, utilizando un hardware especializado como una cúpula de luz o filtros polarizadores. Los enfoques basados en modelos usan la aproximación del modelo de reflectancia de la piel, generalmente en forma de una distribución de reflectancia bidireccional (Función BRDF) o una superficie más compleja bidireccional de dispersión de la función de distribución de reflectancia (BSSRDF). Los impulsado por datos que incluyen los enfoques de Debevec en "Acquiring the reflectance field of a human face",

que fue pionero en la reflectancia de la piel humana de captura en vivo, e introdujo el uso de un cúpula de luz para este propósito. Weyrich en "Analysis of human faces using a measurement based skin reflectance model" también propone el uso de una cúpula de luz, junto con una herramienta de medición de la dispersión del subsuelo con el fin de usar el modelo de la reflectancia de la piel humana para la región facial. Para obtener un formulario de análisis de la BRDF, se ajustan a un Torrance-Sparrow o un Blinn-Phong modelo de reflectancia locales a los datos. Para el BSSRDF, el proceso de instalación utiliza el modelo propuesto por Jensen en "A practical model for subsurface light transport".
A pesar de que estos enfoques presentan resultados verdaderamente realistas, requieren una gran cantidad de información de entrada como parámetro. En las aproximaciones con menores requerimientos de información que utilizan filtros de polarización para la obtención del albedo, se incluye la de Nayaren en "Separation of reflection components using color and polarization". La técnica requiere de dos imágenes de la misma escena, tomada sin ningún tipo de filtros, y una tomada usando un filtro polarizado. La diferencia de las dos imágenes proporciona la reflexión especular del objeto. El principal inconveniente de esta técnica es que las imágenes deben ser tomadas del mismo punto de vista, y hardware especial (los filtros) debe ser empleado. Zickler en "Reflectance sharing: predicting appearance from a sparse set of images of a known shape" amplió la técnica anterior mediante la incorporación de la restricción de que la forma debe ser conocida de antemano. Mediante el uso de la información adicional, que extraen el BRDF.

Utilizando un enfoque PDE, Mallick en "Specularity removal in images and video" recuperó el componente especular de las imágenes, sin el uso de filtros de polarización. Los resultados presentados son convincentes, pero el enfoque es de alta complejidad computacional y no toma en cuenta la superficie real de la cara. Ikeuchi y Sato con "Determining reflectance properties of an object using range and brightness images" propone utilizar el Torrance-Sparrow BRDF al modelo difuso y especular de los componentes de la luz, que ilumina el rostro con el modelo del componente difuso utilizando la formulación Lambertiana. Los parámetros se estiman mediante un ajuste de mínimos cuadrados iterativo, y la distinción entre la difusa especular, y los píxeles de sombra se consigue con un umbral, que se debe establecer manualmente.

De los modelos basados en enfoques como el de París en "Lightweight face relighting", que propuso un enfoque ligero basado en un simple BRDF Phong que utiliza el hardware gráficos. En comparación con el método mostrado en [33], su enfoque tiene tres limitaciones:

1. No se requiere de una sonda de la luz del medio ambiente.
2. La anotación se debe realizar de forma manual.
3. No se ocupa del auto-sombreado.

En este trabajo se muestra que las sombras pueden ser rescatadas sin ningún tipo de artefactos, con un enfoque eficaz y los requisitos de información de entrada incluso más bajo . Blanz y Vetter con "A morphable model for the synthesis of 3d faces", empleó una técnica del modelo morphable para la adquisición de la geometría de las caras de imágenes 2D. El albedo es capturado en el proceso de adaptación. El principal inconveniente es que tienen que inicializar manualmente el modelo de morphable para cada imagen de entrada. En contraste con el método mostrado en [33], es que este es un enfoque estadístico. El carácter descriptivo de los métodos estadísticos depende en gran medida de la variedad y

la calidad del conjunto de entrenamiento y la creación de estos conjuntos no es una tarea trivial. En "Reflectance from images: a model-based approach for human faces" se centran el método para calcular la reflectancia de la cara y también demuestran la transferencia de BRDF. Aunque este trabajo no requiere de un modelo 3D, se toman varias imágenes 2D con las condiciones de iluminación limitada.

Smith y Hancock en "Estimating the albedo map of the face from a single image" presentaron un método para la estimación del albedo de imágenes 2D, que utiliza un modelo de morphable 3D que se utiliza en las imagenes de entrada. Las normales del modelo ajustado se utilizan para el cálculo de la sombra, asumiendo un modelo de reflectancia Lambertiana. La principal limitación de su método es que el supuesto sujeto está iluminado por una sola fuente de luz, que se coloca muy cerca del espectador como se muestra en "Reflectance correction for perspiring faces". Georghiades en "Recovering 3-D shape and reflectance from a small number of photographs" también presentó un método donde sólo las imágenes 2D se utilizan para calcular la piel BRDF y la forma 3D. Este trabajo se limita a las imágenes en escala de grises y no maneja auto-sombreado.

Los métodos anteriormente mencionados sobre el modelado del rostro humano no cuestionan los manejadores-datos (data-driven) de los enfoques en términos de realismo fotográfico. En [33] se propone un método para el modelado de rostro humano y una aplicación para el reiluminado y reconocimiento facial. Donde, un acotado modelo de rostro es fijado en datos crudos 3D usando un marco de trabajo basado en subdivisión del modelo deformable (AFM). Esta representación tiene la geometría y la información de la textura en el mismo espacio de parámetro. Un modelo de análisis de reflectancia de la piel (ASRM) se aplica a la textura para eliminar la iluminación, adquiriendo así el albedo de la cara. A continuación se muestra que los datos obtenidos se pueden utilizar directamente en una variedad de aplicaciones, incluyendo el reconocimiento de rostro. En comparación con otros métodos, el método propuesto ofrece una combinación de características únicas que lo identifica:

1. El método es completamente automático, sus requisitos de entrada son mínimo sólo necesita los datos en 3D y la textura sin la necesidad de realizar las mediciones o calibración de la iluminación del medio ambiente.

2. La representación es compacta y la imagen de la geometría extraída puede ser utilizada directamente en aplicaciones de modelado.

3. La iluminación es eliminada de la textura.

4. El método es altamente eficiente y utiliza la unidad de procesamiento gráfico para acelerar los cálculos de la iluminación logrando una alta eficiencia.

5. Además, el método se puede aplicar a cualquier conjunto de datos faciales ya sea de los escáneres 3D comerciales o bases de datos existentes, lo que permite su amplia utilización en el campo del reconocimiento de rostro humano.

El método propuesto tiene como objetivo obtener la geometría de alta calidad y la textura de los parámetro de entrada mínimo que pueden ser utilizados directamente en varias aplicaciones. En primer lugar, registrar y encajar el modelo de la cara acotada (AFM), utilizando un modelo deformable. Entonces, la geometría se convierte en una representación de la geometría de la imagen que proporciona automáticamente la anotación, el registro y la toma de muestras periódicas. Este trabajo es la continuación de otros

antes mensionados, y elimina la iluminación de la textura mediante un modelo de análisis de reflectancia de la piel (ASRM), adquiriendo así el albedo; teniendo en cuenta que el método propuesto es completamente diferente del modelo de enfoque de morphable Blanz. También tiene como objetivo mostrar que el método propuesto puede mejorar la exactitud de los enfoques vigentes del reconocimiento de rostro. Para esto, se construye una base de datos en 3D y 2D de datos faciales. , en donde los datos 3D únicamente se utilizan para calacular el albedo que sólo se emplea para realizar el reconocimiento 2D de la cara. Este enfoque no es el estado de la técnica en 2D de reconocimiento de rostro, pero puede ser utilizado para evaluar la importancia de eliminar la iluminación, así como las diferencias entre los distintos BRDFs. Un excelente estudio del reconocimiento de la literatura frente 2D/3D es proporcionada por Bowyer en " A survey of approaches and challenges in 3D and multi-modal 3D + 2D face recognition".

La deducción de un denso modelo 3D de la superficie de una cabeza humana de una secuencia de video monocular es un problema muy difícil. T. Akimoto en "Automatic creation of 3D facial models" crea un sólo modelo 3D de la cabeza de una vista frontal y una vista de perfil mediante el ajuste de un modelo genérico. Tang y Huang en "Automatic construction of 3D human face models based on 2D images" crean un modelo 3D por la ubicación de 32 puntos característicos de la cara en múltiples imágenes y relacionarlas con los nodos de un modelo genérico de la cara. Este método crea un modelo 3D razonable, pero no es lo suficientemente denso para representar la forma de la cara con precisión.

Fua en "Using model-driven bundle-adjustment to model heads from raw video sequences" utiliza un modelo impulsado por el método de ajuste de paquete para obtener el modelo de la cabeza de una secuencia de vídeo en bruto. Aprovecha los datos de los bordes de la silueta, y una función de los puntos 2D. R. Lengagne en "3D face modeling from stereo and differential constraints" deriva un modelo 3D de la cara estéreo y restricciones diferenciales. Estas limitaciones son útiles en la recuperación de la superficie, en particular, en las zonas donde los métodos de estéreo tienden a fallar. Y. Li en "Modelling faces dynamically across views and over time" ha propuesto un escaso 3D, punto de distribución modelo creado a partir de imágenes 2D en diferentes puntos de vista con el fin de asignar un modelo 3D simplificado de la malla de la cara en el 2D capturado de imágenes tridimensionales. La malla obtenida es muy escasa y no es adecuada para la caracterización de las características 3D faciales.

En "Model-based bundle adjustment with application to face modeling", los autores proponen un modelo basado en el algoritmo de ajuste de paquete que tiene un conjunto de pistas de imagen de entrada, sin un previo modelo 2D a 3D de asociación. El uso de un modelo genérico de la cara permite la reducción de la complejidad del algoritmo de ajuste de paquete, pero restringe el uso de la superficie 3D obtenida para la identificación. Las características del modelo 3D de la cara se proyecta sobre el modelo genérico y por consiguiente, ignora las características muy específicas que permiten la identificación de una persona de sus características 3D. También vale la pena destacar que el enfoque adoptado por Romdhani en "Face identification by fitting a 3D morphable model using linear shape and texture error functions", que genera un modelo 3D de una sola imagen. Si bien los resultados son visualmente agradables, la precisión métrica debe establecerse, en particular, dado el modelo de proyección ortográfica utilizada. Por otra parte, este método requiere un

manual de paso crítico para iniciar el proceso. A continución se mostrará con más detalle el método propuesto por Gerard Medioni en [25] que ha sido el trabajo más reciente que le da continuidad a este estudio.

En [25] se presenta un método para identificar a los individuos no cooperativos a distancia usando una secuencia de imágenes y un modelo 3D del rostro. La mayoría de los elementos biométricos (como huellas digitales, forma de la mano, el iris, la retina o las exploraciones) requieren características de muy estrecha cooperación en proximidad al sistema biométrico. En la implementación se procesa las imágenes obtenidas con una cámara de ultra-alta resolución de video, donde se infiere la ubicación de la cabeza de los sujetos. Esta información se utiliza para los recortes de la región de interés y construir un modelo 3D del rostro para realizar la identificación biométrica. El experimento se lleva a cabo en modelos 2D y 3D de bases de datos. A continuación se muestran las principales características que describen al sistema propuesto:

1. Se encuentran métricas del modelo 3D facial que pueden ser usadas para el reconocimiento de la escala, utilizando el método sencillo, aunque no existe una escala exacta en la reconstrucción 3D.

2. Los experimentos se hicieron con un motor 3D comercial que indica la viabilidad del enfoque propuesto para el reconocimiento en contra de galerías 3D, a una distancia (3, 6 y 9 m).

3. Se demuestra frente a los resultados de modelos 3D de varios factores, incluyendo el movimiento de la cabeza, las condiciones de iluminación exterior, y las gafas. Los resultados de la evaluación indican que los datos de video sólo a una distancia de 3 a 9 metros, puede proporcionar una forma de la cara 3D que apoya al reconocimiento de la cara con éxito.

Este artículo usa los datos biométricos para identificar a las personas que tienen las siguientes características: deben ser universales, únicas (que permiten la discriminación entre un gran número de individuos), y de alta permanencia (que no cambian con el tiempo, la edad, el estado emocional, etc). También debe permitir recoger de forma fácil los datos para utilizar estos elementos biométricos. Muchas de las características que son muy distintivas requieren un tema de cooperación en las proximidades del sistema biométrico. Tales características no son utilizables cuando se tiene que tratar con un individuo no cooperativo que se quiera observar discretamente y a distancia, como se requiere para las aplicaciones de seguridad en muchos casos. El objetivo de este artículo es llevar a cabo un reconocimiento facial en 3D mediante un modelo de cara 3D generada a partir de una secuencia de imágenes a distancia.

El reconocimiento tridimensional es una alternativa atractiva para el reconocimiento 2D bajo desafiantes condiciones de ambientes como las variaciones de la pose y los cambios de iluminación. Uno de los métodos utilizados en el reconocimiento de rostro en este artículo usa un modelo 3D para probar un rostro que hace juego contra una base de datos 3D cuándo sólo una galería 3D está disponible. Otro método, dado un preciso modelo 3D afronta una imagen poco frontal de la cara que puede ser transformada en una cara frontal. Esto es imposible de hacer exactamente usando una imagen simple que distorsiona las técnicas sin conocimiento de una rostro en 3D. Los métodos múltiples de reconocimiento aumentan la probabilidad de éxito bajo una variedad amplia de condiciones de operación.

Este trabajo [29] presenta una aplicación de red neuronal llamada Neocognitron , usando una arquitectura de computación de alto rendimiento basada en la GPU. Neocognitron es una red neuronal artificial, propuesto por Fukushima y colaboradores, constituido por varias etapas jerárquicas de las capas de neuronas, organizadas en matrices de dos dimensiones denominados planos celular. Para el cálculo de alto rendimiento de la aplicación de reconocimiento facial mediante Neocognitron fue utilizado CUDA. Neocognitron es una red neuronal masivamente paralela, compuesta por varias capas de células neuronales. Neocognitron pueden ser procesados en dos fases:

1. Fase de formación auto organizada (self-organized)

2. Fase de reconocimiento

Con esta aplicación se alcanzó un aumento significativo en el rendimiento del procesamiento del reconocimiento de la cara neocognitron, demostrando la viabilidad de utilizar este método. Sin embargo, el tamaño de las imágenes utilizados en la operación eran pequeñas, 57 x 57, esto permitió la carga completa de la estructura de la red en la memoria del dispositivo donde el acceso está protegido y es de alta velocidad. Estos condiciones influyeron mucho en los resultados obtenidos por este trabajo. La implementación del algoritmo para ser ejecutado en la GPU se realizó con la arquitectura CUDA que le dio un aumento de una super-velocidad lineal-up ($SP > p$) . Esto probablemente se produjo por las diferencias en la arquitectura. Por otra parte se observó un alto rendimiento si se compara el tiempo de procesamiento. Otra de las conclusiones sobre la aplicación de este proyecto, es que esto minimiza algunos de los problemas comunes existentes, cuando se utilizan otros entornos de computación paralela:

1. Sincronización: desde la granularidad de desarrollo dentro de este dispositivo no se está compitiendo por los recuerdos compartidos (cada hilo tiene su punto/área de la memoria) donde hay la necesidad de una pérdida de tiempo para lograr una sincronización de los procesadores

2. Red: si todo el proceso ocurre dentro del mismo dispositivo existe el problema del tipo de conexión y la velocidad de toda la arquitectura de la GPU

3. Contención: no hay competencia de los recursos por los procesadores.

Este trabajo [34], aunque no esté en ninguna de las ramas de la taxonomía mostrada es de gran interés para futuros trabajos por los aportes que realiza. Presenta una serie de modificaciones a la biblioteca GPUCV que le permiten ser utilizado de forma nativa en la plataforma Linux y así lograr las ventajas del rendimiento proporcionado por las GPUs para aplicaciones de visión por computadora que se ejecuta en Linux. Las decisiones involucradas en este proceso se muestran en este artículo y además se discuten las características de funcionamiento de la biblioteca GPUCV en esta nueva plataforma comparándola con la plataforma original de Microsoft Windows. Hubo varios factores que motivaron a los autores a realizar la implementación de la biblioteca GPUCV para la plataforma Linux.

Entre estos tenemos que la biblioteca GPUCV tiene estrechos vínculos con la biblioteca OpenCV, que está disponible en una amplia variedad de plataformas, por lo que es útil para GPUCV también ser multiplataforma con el objetivo de que las aplicaciones existentes de OpenCV puedan aprovechar la aceleración de la GPU, utilizando simplemente el procesamiento de operadores de imagen GPUCV en lugar de los equivalentes OpenCV. Llevar la biblioteca GPUCV a las plataformas soportadas por OpenCV ayuda a lograr

este objetivo más plenamente. Este informe ha estudiado el papel y la importancia de la GPU en la aceleración de algoritmos de visión por computador y luego se centró en llevar estas ventajas a la plataforma Linux a través de la biblioteca GPUCV. Una versión nativa de Linux de esta biblioteca fue creada y su desempeño fue punto de referencia y de comparación a la versión original de Windows. Mientras que el rendimiento de la biblioteca GPUCV bajo Linux sea algo menor que en Windows, las ventajas de rendimiento presentes en GPUCV aún representan una mejora significativa en el rendimiento de los algoritmos de visión por computadora en la plataforma Linux.

6. Conclusiones

Con este estudio podemos llegar a la conclusión, que dentro del campo del reconocimiento de rostros, desafortunadamente, los mejores sistemas de reconocimiento en 2D que enfrentamos hoy en día no son fiables ni lo suficientemente exactos para la iluminación arbitraria y se plantean en ambientes sin restricciones. Muchos sistemas de reconocimiento automático de rostro han sido desarrollados y se le han realizado diversas pruebas en el campo que demuestran que los métodos de reconocimiento facial mediante imágenes son sensibles a los parámetros del medio ambiente, tales como: las condiciones de iluminación y la pose del sujeto. Incluso bajo condiciones favorable (control de iluminación, buena calidad de imagen, resolución suficiente, las imágenes frontal y expresión neutral), los mejores algoritmos en estos casos producen en la mayoría una tasa de error de 6 %, y por lo tanto, el rendimiento es poco probable que sea suficiente para la mayoría de las aplicaciones. Estas pruebas de campo de los sistemas en condiciones realistas y operacionales muestran un rendimiento reducido, obteniendo resultados inaceptable para los mejores sistemas de control de procesos automatizado.

El reconocimiento facial tridimensional es objeto de atención considerable para futuras investigaciones, ya que se piensa comúnmente que el uso de una forma 3D podría superar las limitaciones fundamentales de reconocimiento 2D. Las principales ventajas del uso de 3D para el reconocimiento se muestran en la variación y compensación de la iluminación. La postura de las variaciones pueden ser resueltos por la alíneación en 3D, y las variaciones de iluminación no afecta a los modelos 3D. Parece que el reconocimiento de rostro en 3D especialmente en combinación con 2D, es una promesa importante, y que podría alcanzar una precisión comparable con otros datos biométricos como las huellas dactilares y del iris. La inferencia de un modelo denso de la superficie 3D de una cabeza humana de una secuencia de video monocular es un problema muy difícil de la visión del computador, para el cual no existe ninguna solución en la bibliográfia consultada, y es un tema muy interesante para futuras investigaciones.

Otra observación es, que los algoritmos dedicados a esta rama implementados en la unidad de procesamiento gráfico son mucho más eficientes que sus homólogos en la CPU, señalando que muy pocos de ellos están diseñados para ejecutarse en una GPU ATI o en otro sistema operativo que no sea MS Windows. Para futuras investigaciones se destaca en el campo del reconocimiento de rostros en cuanto a lo práctico, el desarrollo de aplicaciones de este ámbito que utilicen una arquitectura multiplataforma a nivel de hardware (Fabricadas

por las empresas Nvidia y ATI) y software. Utilizar un sistema embebido con framebuffer usando un núcleo en tiempo real permitiría notablemente un aumento de rendimiento aparte del obtenido con la aceleración del algoritmo implementado en la GPU. En cuanto a lo teórico una interesante línea podría ser el estudio de los modelos faciales como las características faciales de los ojos y la boca que necesitan separarse de los modeles de geometría y reflectancia, prestándole atención al efecto del pelo en el rostro.

Para futuras investigaciones sobre el mejorar la eficiencia de los algoritmos implementados para ser ejecutados en la GPU, se destaca el uso de la memoria compartida de cada multiprocesador como uno de los programas gestionado por la cache y disminuir todo lo posible la latencia de las cargas repetitivas al sistema de memoria global de estos dispositivos.

Referencias bibliográficas

1. *NVIDIA CUDA Compute Unified Device Architecture-Programming Guide*, 2007.
2. Floating-point operator v4.0 datasheet. April 2008.
3. C graphics, http://libsh.org/, Noviembre 2009.
4. Microsoft high shading language, Noviembre 2009.
5. Opengl shading language, http://www.opengl.org/documentation/oglsl.html, Noviembre 2009.
6. Sh, http://libsh.org/, Noviembre 2009.
7. Tasora A., Negrut D., and Anitescu M. Large-scale parallel multi-body dynamics with frictional contact on the graphical processing unit. 2008.
8. Analysis, Patrick S. Mccormick, Jeff Inman, and James P. Ahrens. Abstract scout: A hardware-accelerated system for quantitatively driven visualization.
9. He B., Yang K., R Fang, Lu M., Govindaraju N. K., Luo Q., and Sander P. V. *Relational joins on graphics processors*. 2008.
10. Wu B. and Nevatia R. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. 2007.
11. Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3d morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1063–1074, 2003.
12. Feng Cao, Anthony K. H. Tung, and Aoying Zhou. Scalable clustering using graphics processors, 2006.
13. Lieberman M. D., Sankaranarayanan J., and Samet H. *A fast similary join algorithm using graphics processing units*. 2008.
14. Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *In CVPR*, pages 886–893, 2005.
15. Jérémie Detrey and Florent de Dinechin. Parameterized floating-point logarithm and exponential functions for fpgas. *Microprocess. Microsyst.*, 31(8):537–545, 2007.
16. Ian Foster. *Designing and Building Parallel Programs*. 2003.
17. Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Generative models for recognition under variable pose and illumination. pages 277–284, 2000.
18. Naga K. Govindaraju, Jim Gray, Ritesh Kumar, and Dinesh Manocha. Gputerasort: High performance graphics coprocessor sorting for large database management. pages 325–336. SIGMOD, 2006.
19. Kazuhiro Hilada. A real-t ime face tracking system based on morphable 3d model fitting. 2003.
20. IBM. *Cell SDK 3.1*, 2008.
21. Bernstein D. J., Chen T. R., Cheng C. M., Lange T., and Yang B. Y. *Ecm on graphics cards*. 2009.
22. Michael Jones, Paul Viola, Paul Viola, Michael J. Jones, Daniel Snow, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *In ICCV*, pages 734–741, 2003.
23. K. Chang K., K. Bowyer, and P. Flynn. Face recognition using 2d and 3d facial data. pages 25–32, 2003.
24. Nachiket Kapre and A. Dehon. Performance comparison of single-precision spic model-evaluation on fpga, gpu, cell, and multi-core processor. 2009.

25. Gerard Medoni. Identifying noncooperative subjects at a distance using fce images and inferred three-dimensional face models. 39, January 2009.
26. P. J. Mucci, S. Browne, C. Deane, and G. Ho. Papi: a portable interface to hardware performance counters. pages 7–10, 1999.
27. Bohm C. Noll, R. Plant C., and Zherdin A. *Index supported similarity join on graphics processors.* 2009.
28. Sabzmeydani P. and Mori G. Detecting pedestrians by learning shapelet feature. 2007.
29. Gustavo Poli, Jose Hiroki Saito, Joao F. Mari, and Marcelo R. Zorzan. Processing neocognitron of face recognition on high performance environment based on gpu with cuda architecture. 2008.
30. Manavski S. and Valle G. Cuda compatible gpu cards as effecient hardware accelerators for smith-waterman sequence alignment. 2008.
31. Arul Shalom, Manoranjan Dash, and Minh Tue. *Efficient k-means clustering using accelerated graphics processors.* 2008.
32. Sudipta N. Sinha, Jan michael Frahm, Marc Pollefeys, and Yakup Genc. Gpu-based video feature tracking and matching. Technical report, in Workshop on Edge Computing Using New Commodity Architectures (EDGE 2006), Chapel, 2006.
33. George Toderici, Georgios Passalis, Theoharis Theoharis, and Ioannis A. Kakadiaris. An automated method for human face modeling and relighting with application to face recognition. March 2008.
34. Richard Viney and R. Green. Gpu-accelerated computer vision on the linux platform. 2007.
35. Liu W., Schmidt B., Voss G., and Muller Wittig W. *Molecular dynamics simulations on commodity gpus with cuda.* 2007.
36. Li Zhang and R. Nevatia. Efficient scan-window based object detection using gpgpu. 2008.

RT_031, junio 2010
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: DCG Matilde Galindo Sánchez
RNPS No. 2142
ISSN 2072-6287
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba