Proyecto: Calculo Paralelo en el seguimiento visual de objetos en imágenes

<<Pintos,Leandro[1], Porro,Diego , Abdala, Nahuel , D'ortona,Agustín , Cáceres,Silvina>>

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
<<leopintos11@gmail.com, pdiego93@gmail.com, Nahuel.abdala@gmail.com, agustin.dortona@gmail.com, silvinacace@gmail.com>>

Resumen

El presente trabajo tiene como propósito extender las funcionalidades de los trabajos prácticos realizados sistemas embebidos y Android,adicionando una nueva funcionalidad que incorpora el procesamiento de un complejo algoritmo solicitando el uso de computación de altas prestaciones (HPC).Para poner a prueba esta funcionalidad se dispone de una cámara que permite realizar el procesamiento de imágenes obtenidas del entorno, las cuales se aplican a la tecnología GPU(Unidad de procesamiento gráfico), En este paradigma la GPU permite la programación paralela y el lenguaje original para programar es CUDA(Arquitectura Unificada de Dispositivos de Cómputo), es una variación del lenguaje de programación C, sin embargo es posible utilizar otros como C++,Python,Fortran,Java y Matlab.

En esta investigación nos enfocamos en CUDA C para desarrollar las funciones que se ejecutan en la GPU y Matlab ya que facilita la migración a la GPU y además ofrece la posibilidad de utilizar código C.

Palabras claves: Imágenes,Processamiento paralelo,arquitectura CPU, arquitectura GPU, tecnología CUDA, pixel, algoritmo,rendimiento.

Introducción

La extensión funcional aplicada consiste en agregar una cámara sobre la plataforma que capture imágenes del entorno realizando un seguimiento de los objetos. La finalidad es la estimación de la posición de los objetos en imágenes a través de una cámara de video.

Es un proceso complejo que engloba una multitud de técnicas de visión de alto costo computacional. Una vez localizado el objeto hay que prestar atención a una serie de características visuales que permitan realizar el seguimiento, estas características suelen ser el color, la textura o el flujo óptico. El flujo óptico es el patrón de movimiento aparente entre los objetos causado por el movimiento relativo entre el observador y la escena. En la actualidad el cálculo del flujo óptico se utiliza en el problema de seguimiento de objetos, detección de movimiento, estabilización de video, esxtracción de fondo, odiometría visual, y compresión de video entre otras.

Estado del Arte

Definiciones teóricas

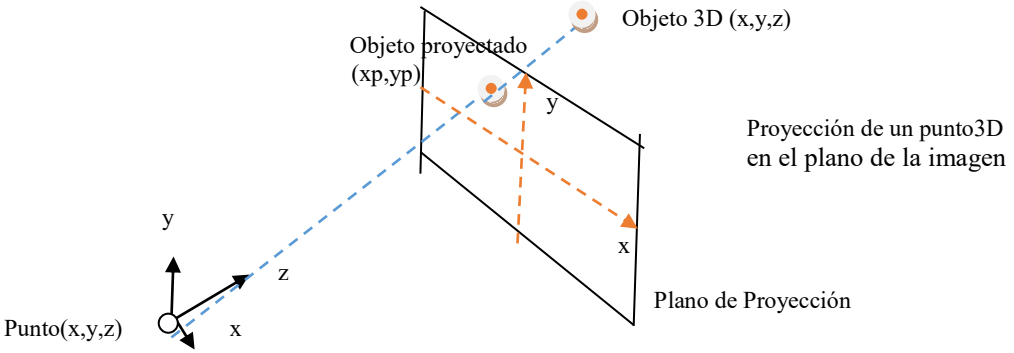
GPU(Unidad de Procesamiento Gráfico): Dividida en distintas unidades funcionales, se ejecuta código a ser paralelizado permitiendo velocidades de cómputo muy superiores,y así optimizar el rendimiento computacional.

CUDA(Arquitectura Unificada de dispositivo de cómputo): Desarrollada por NVIDIA, mejora las funcionalidades incorporando a cada unidad una unidad aritmético lógica (ALU) y una memoria para almacenar el cache necesario para la ejecución de los programas, desarrollando las llamadas GPGPU o GPU de propósito general.

Matlab: Es un lenguaje de programación de alto nivel orientado al cálculo numérico, visualización y programación. Permite trabajar de manera interdisciplinar gracias a la cantidad de herramientas que ofrece como puede ser procesamiento de señales e imagen, comunicaciones,sistemas de control,etc.

Desarrollo

Cuando tratamos con imágenes estamos trabajando sobre proyecciones en dos dimensiones de un entorno tridimensional, es decir, no podemos captar toda la información completa. Dada una secuencia de imágenes podemos estimar la proyección de un movimiento tridimensional sobre el plano de la imagen del observador utilizando diversas técnicas.Como producto de estos tratamientos se puede obtemer el campo de velocidades y los desplazamientos de los pixeles dentro de la imagen, esta información puede utilizarse para recomponer el movimiento real del observador así como para detección de objetos, segmentación de objetos, y detección de colisiones, entre otras muchas aplicaciones más.



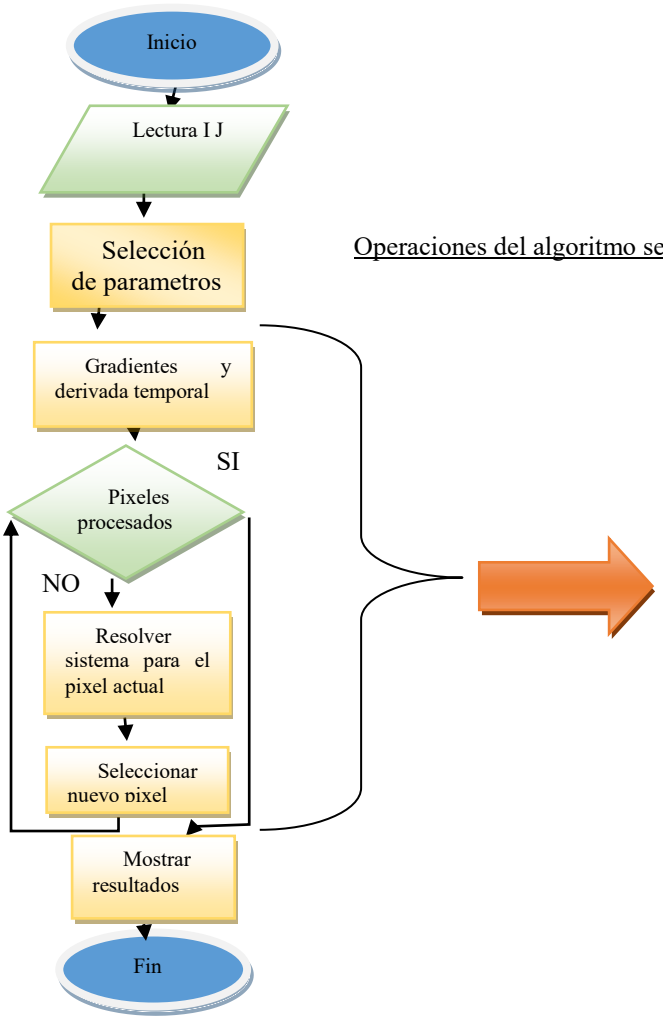
Una de las opciones para la obtención del flujo óptico es realizar ciertos cálculos de aproximación bastante buena, de una secuencia de imágenes se seleccionan dos consecutivas a las cuales se les realiza un tratamiento visual y una serie de operaciones para obtener el resultado final y mostrarlo.Repitiendo el tratamiento sobre los distintos pares se obtiene un análisis completo de todo el video.

Para obtener los datos necesarios para resolver el problema se diseñó un algoritmo que utiliza las funcionalidades que ofrece el cálculo paralelo. El algoritmo procesa un tipo de imagen muy concreta, imágenes en escala de grises o imágenes de intensidad lumínica. Una imagen digital en escala de grises es una imagen en la que cada pixel toma un valor comprendido entre 0 y 255.Este valor está relacionado con la medida de intensidad que captura el sensor de la cámara para un determinado rango del espectro electromagnético, es decir, se captura la intensidad lumínica de un color. A la hora de realizar una captura fiel a la realidad, las cámaras recogen la información de luminosidad de distintas frecuencias para recomponer la imagen siguiendo una serie de modelos o formatos, entre ellos se destaco el modelo RGB.
El algoritmo requierer del siguiente procesamiento:
Selección de pixeles: el algoritmo está enfocado a obtener el desplazamiento de los pixeles de la imagen.Para ello es necesario seleccionar que pixeles queremos tratar.
Obtención de derivadas temprales y gradientes: Estas operaciones arrojan información necesaria sobre la que se aplica la formulación.Realizan el fuerte cálculo matamático. Las operaciones a realizar son: Cálculo de los gradientes y cálculo de la derivada temporal .
Método de los mínimos cuadrados: Cuya aplicación permite obtener los valores del desplazamiento de los píxeles de la imagen. Se realizan operaciones de sistemas de ecuaciones.
Todas estas operaciones se tratan en una arquitectura HPC, haciendo provecho del uso de GPU y la programación paralela obteniendo un óptimo rendimiento, ya que el procesamiento de imágenes requiere de una complejidad de cálculos computacionales que la tecnología GPU es capaz de ofrecer en la optimización y aceleración de algoritmos y reducción en los tiempos de computación.
.

Explicación del algoritmo

Las entidades principales sobre las que estamos trabajando son mágenes , concretamente sobre los valores de intensidad de sus pixeles. Muchas de las operaciones que se están aplicando consisten en repetir ciertas modificaciones variando unicamente los datos de partida, ejecutamos el mismo código sobre distintos píxeles. Para la ejecución secuencial este proceso se vale de una serie de bucles que indexan los valores de las matrices , en el caso de la programación paralela podemos ejecutarla de manera parlela. Por

consiguiente , la mayor parte del tratamiento de imágenes y el gran procesamiento de cálculo matemático se pueden migrar a la GPU.



Operaciones del algoritmo secuencial que pueden paralelizarse

Operaciones en la GPU
Obtención de la matriz de intensidad/conversion RGB-gris (Tratamiento de la imagen)
Obtención de los gradientes
Obtención de las derivadas
Aplicar Sistemas de Ecuaciones

Como se van a ejecutar programas en la GPU se necesitan configurar ciertos parametros en comparación con el procedimiento en la CPU:
Numero de threads: Se seleccionan tantos hilos como píxeles contenga la imagen, siempre que la GPU pueda procesarlo. En el caso de que el número de pixeles sea superior a la capacidad de cómputo paralelo se irán ejecutando por tandas.
Una vez configurado el número de hilos necesarios ejecutan paralelamente en la GPU cada uno de los kernels que implementan las operaciones: Obtención de la matriz de intensidad/conversión RGB-gris, Obtención de los gradientes, Obtención de las derivadas, Aplicar Sistemas de Ecuaciones.
El kernel es la función que se ejecuta de manera masiva en el GPU, esta función se invoca desde la CPU la cual delega el trabajo en la GPU, creando múltiples hilos encargados de ejecutar el kernel de manera individual e independiente.

Se establece una diferencia entre el algoritmo secuencial y el paralelo, mientras que el algoritmo en la CPU debe esperar la resolución de un pixel para continuar con el siguiente, en el algoritmo que utiliza la GPU se resuelven todos los sistemas de manera paralela. Por consiguiente, el objetivo de cada hilo es realizar la misma operación que el algoritmo en la CPU realizaba en cada vuelta de bucle, resolver el sistema de ecuaciones para cada pixel.
La información que recibe cada hilo será la necesaria para la resolución del sistema de ecuaciones, gradientes, derivadas y localización del pixel sobre el que se va a trabajar.
A continuación se describe un pseudocódigo que resume a grandes rasgos la estructura que tendrá el programa una vez implementado en C.

Resolución del sistema, donde x=desplazamiento :Ax=b

```
Resuelve_sistemas_de_ecuaciones(Ix,Iy,It,w,qi,qj)
{
 Idx←Obtener_Identificador_Hilo();
 Asociamos el pixel al hilo, obtención de la fila: f=qi(Idx);
 Asociamos el pixel al hilo, obtención de la columna: c=qj(Idx);

 For i=f-w/2 to f +w/2 do          }   Se recorre la vecindad del punto (f,c)
   For j=c-w/2 to c+w/2 do         }

      Ex=Ex+Ix(i,j)^2              }   Información referente a la matriz A:  A=[ Ex   Exy ]
      Ey=Ey+Iy(i,j)^2              }                                         [ Exy  Ey  ]
      Exy=Exy+Ix(i,j) Iy(i,j);     }

      b1=b1 +Ix(i,j) It(i,j);      }   Vector b del sistema Ax=b
      b2=b2 + Iy(i,j) It(i,j);     }

   End for
End for
```

Pruebas que pueden realizarse

La idea principal es simple, dado dos imágenes pertenecientes a una secuencia de movimiento continuo, el método es capaz de identificar el desplazamiento de los píxeles en cada frame y de mostrarlo en forma de vector.
Como ejemplo se parte de dos matrices de nxm elementos que simulan las medidas de intensidad de dos imágenes consecutivas y se obtendrá el desplazamiento del pixel correspondiente a una determinada coordenada(ni,mj). Por ejemplo. Si se tienen dos matrices de 6x6 elementos, se puede obtener el desplazamiento en el espacio del pixel perteneciente a la coordenada(2,2) hasta el punto de coordenada (2,3), es decir, un desplazamiento horizontal de un pixel.

Conclusiones

La aplicación de estos programas optimizados está enfocada a ser aplicada a un problema concreto, para el caso propuesto, el procesamiento de imágenes. El algoritmo presentado ofrece buenos resultados a la hora de calcular el flujo óptico y desplazamiento entre frames de una secuencia, y gracias a la pararelización usando la GPU, la mejora del rendimiento es notable.
Con respecto a los lenguajes de programación en los que es posible implementar el algoritmo diseñado es de gran ayuda para el programador programar en Matlab utilizando las bibliotecas que ofrece ya que permite unir de manera eficiente el tratamiento de imágenes junto con la programación paralela, como también programar en C siempre que las bibliotecas fuesen insuficientes.
Si en un futuro se desea implementar el algoritmo en desarrollo de aplicaciones para Android es posible ya que se puede trabajar en la GPU desde Java mediante el uso de las librerías “jCuda” que conecta Java con la GPU, como también es posible importar Matlab dentro de Android Studio.

Referencias

González Iturbide ,José Jaime: Optimización de algoritmos para procesado de imágenes con GPUs : Universidad Politécnica de Madrid. 21..23, 35..44 (2010)

Yurivilca, Cristian Tinoco : Implementación de un algoritmo basado en el dominio de las frecuencias y procesamiento en paralelo para la generación de una imagen médica en super resolución. Universidad Nacional de Ingeniería de Lima. 20..31, 37..45 (2014)

Bermúdez, Enrique Oriol : Algoritmos de reconstrucción de Escenas 3D paralelizados en GPU para aplicaciones en tiempo real. Universidad Politécnica de Catalunia. 13,14,37,51 (2010)

Cano Ocampo, Sergio Alberto , Díaz López, Neilor Esteban : Implementación de un algoritmo para eliminación de ruido impulsivo en imágenes y análisis comparativo de tiempos de respuesta bajo arquitectura GPU y CPU. Universidad de Medellín Facultad de Ingeniería. 37..45 (2013)

Procesamiento Paralelo CUDA, https://www.nvidia.es/object/cuda-parallel-computing-es.html

Computación acelerada por GPU, https://la.nvidia.com/object/what-is-gpu-computing-la.html

Importar proyecto dentro de Google Android Studio,
https://www.mathworks.com/help/supportpkg/android/ug/export-project-to-google-android-studio.html