Proyecto: Cálculo 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 la funcionalidad del sistema embebido realizado, analizando la utilización de computación de altas prestaciones (HPC) para poder procesar algoritmos de procesamiento de imágenes. Para poner a prueba esta funcionalidad se dispone de una cámara que obtiene 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), el cual 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, paralelismo, CPU, GPU, CUDA C, pixel, algoritmo, rendimiento.

Introducción

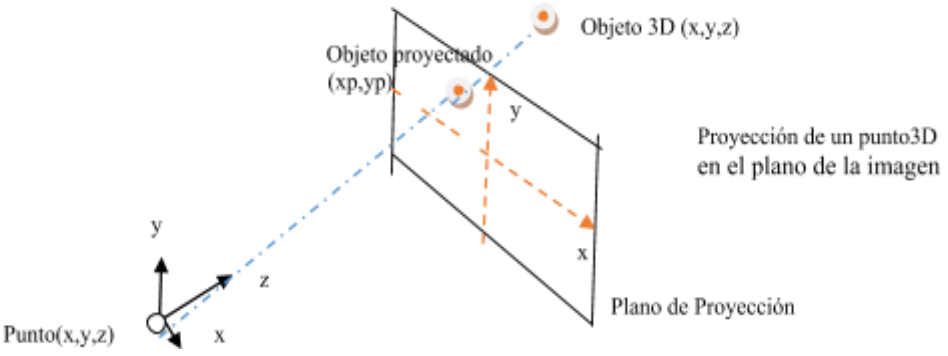
La presente investigación consiste en adicionar una nueva modalidad de trabajo al sistema embebido actual.

El presente sistema consiste en una plataforma que puede funcionar como mesa de trabajo, la cual se autonivela automáticamente para mantenerse siempre pareja incluso reaccionando en tiempo real a cambios de niveles en la superficie donde se encuentra apoyada. Dicha función se complementa con una aplicación Android, la cual permite controlar el modo de trabajo de la plataforma mediante comunicación Bluetooth con la misma.
Dado de que la plataforma puede ser utilizada en diferentes modos, puede surgir la necesidad de que funcione siempre con una inclinación determinada. Por ejemplo, si se utiliza como mesa de lectura, siempre deberá tener una ligera inclinación hacia donde se encuentra el lector para facilitar su comodidad en la lectura. Esto sería muy útil en un entorno de constante cambio de niveles de inclinación, como por ejemplo: a bordo de un barco.
La extensión funcional aplicada consiste en agregar pequeñas cámaras a la plataforma que capturen imágenes del entorno detectando de qué lado se encuentra el lector o usuario. La finalidad es la estimación de la posición del usuario y en base a esto, mantener la inclinación fija hacia esa dirección.
El procesamiento de imágenes requiere de ejecución de operaciones en paralelo. Ésto lo permite la arquitectura GPU obteniendo velocidades de cómputo muy superiores y así optimizando el rendimiento computacional en comparación con la utilización de CPU secuencial.

Actualmente, esta capacidad de utiliza para diversas aplicaciones, como por ejemplo navegación autónoma o asistida en vehículos, control y monitoreo en áreas públicas, reconocimiento de objetivos y análisis del comportamiento. La masificación del acceso a internet y el advenimiento de las redes sociales ha desembocado en la creación de inmensos repositorios de imágenes y videos naturales, como pueden ser Facebook, Youtube, Snapchat, Instagram, etc. Esto ha dado lugar a nuevos requerimientos de escala y robustez para los algoritmos de visión por computadora como así también a nuevas aplicaciones basadas en el procesamiento y análisis masivo de datos. Como respuesta a estos nuevos factores se han ido incorporando cada vez más el uso de nuevas técnicas relacionadas a aprendizaje automático y reconocimiento de patrones, para poder analizar de manera automática estos grandes conjuntos de datos.

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 obtener el campo de velocidades y los desplazamientos de los píxeles 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.



Para el procesamiento de las imágenes utilizaremos la técnica de flujo óptico. Una de las opciones para la obtención del flujo óptico el el llamado algoritmo de Lucas-Kanade(KL) que permite 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 distintos pares de imágenes se obtiene un análisis completo de todo el video. El algoritmo procesa un tipo de imagen muy concreta, imágenes en escala de grises o imágenes de intensidad lumínica.

A la hora de realizar una captura fiel a la realidad, la cámara recoge la información de luminosidad de distintas frecuencias para recomponer la imagen siguiendo una serie de modelos o formatos, entre ellos se destaca el modelo RGB(Red Green Blue).

El algoritmo Lucas-Kanade(KL) requiere del siguiente procesamiento:

Selección de píxeles: el algoritmo está enfocado a obtener el desplazamiento de los píxeles de la imagen.Para ello es necesario seleccionar qué pixeles queremos tratar.

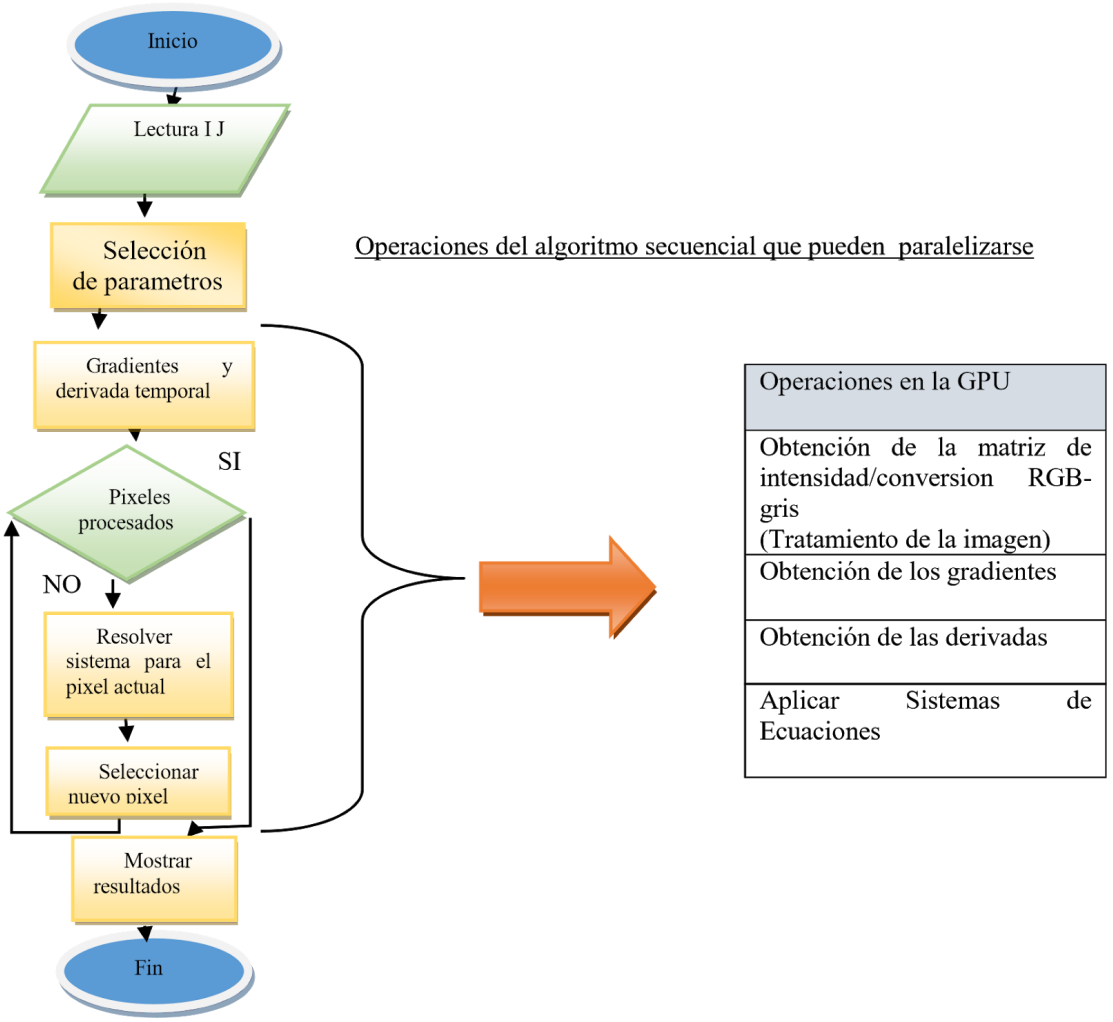
Obtención de derivadas temporales y gradientes: Estas operaciones arrojan información necesaria sobre la que se aplica la formulación.Realizan el fuerte cálculo matemático.

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. Para desarrollar esta implementación paralela se utiliza el lenguaje original de GPU que es CUDA(Arquitectura unificada de dispositivos de cómputo) y es una variación del lenguaje C, como también haremos uso de las librerías

JCuda que conecta Java con la GPU, ya que nuestra aplicación de dispositivo android es compatible con Java.

Explicación del algoritmo

Las entidades principales sobre las que estamos trabajando son imágenes , concretamente sobre los valores de intensidad de sus píxeles. Muchas de las operaciones que se están aplicando consisten en repetir ciertas modificaciones variando únicamente 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 paralela. Por consiguiente , la mayor parte del tratamiento de imágenes y el gran procesamiento de cálculo matemático se pueden migrar a la GPU.



Como se van a ejecutar programas en la GPU se necesitan configurar ciertos parámetros en comparación con el procedimiento en la CPU:

Número 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 píxeles sea superior a la capacidad de cómputo paralelo se irán ejecutando por tandas.

Por ejemplo, para una imagen de 1024x768 pixeles (786.432 pixeles en total) en una GPU que tiene capacidad para correr 4096 hilos en paralelo:

786.432px/4096 threads = 192 tandas.

Una vez configurado el número de hilos necesarios ejecutan paralelamente en la GPU cada uno de los kernels que implementan las operaciones, se obtiene 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
```

Para la implementación paralela del algoritmo de LK determinamos realizar dos funciones: LK_main y LK_GPU.

La función LK_main deberá realizar: Lectura de imágenes, tratamiento de imágenes y obtención de los valores de intensidad y número de píxeles a procesar.

La función LK-GPU deberá realizar el procesamiento combinado en la GPU.

Programa Paralelo:

LK_main: [u,v]= LK_GPU(imag1,imag2,ww,'grid')

Lk_kernel.cu

LK_main: Encargada de iniciar las variables, cargar imágenes y mostrar resultados , llama a la función LK_GPU la cual devuelve el valor del flujo óptico utilizando una llamada a un kernel en la GPU.

LK_GPU: Configura y ejecuta un kernel, lk_kernel.cu que procesará la información en la GPU.
Lk_kernel.cu: kernel ejecutado en la GPU.

Pruebas que pueden realizarse

Las pruebas a realizarse consisten en ir variando la cantidad de pixeles a tomar por cada muestra de imagen para verificar cuántos píxeles se necesitan para cumplir con la detección del usuario correctamente. Por otra parte, también probar el comportamiento del algoritmo ante una imagen con poca variación de colores, ya sea por iluminación (poca iluminación genera oscuridad, pero mucha iluminación puede generar encandilamiento), o por el propio entorno, en caso de que el usuario y la habitación tengan colores similares.

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 ventajas a la hora de calcular el flujo óptico y desplazamiento entre frames de una secuencia y, gracias a la paralelización utilizando la GPU, la mejora del rendimiento aumenta.
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

1. Callejas Ramos,Alejandro Iván.: Algoritmos de reconstrucción de imágenes de escala de grises mediante GPU . (2016)
2. Hernandez Anillo, Luis Felipe.: Desarrollo de aplicaciones con técnica de programación paralela para el análisis del procesamiento 3D de imágenes de microscopía. Universidad Tecnológica de Bolívar, Programa de Ingeniería de Sistemas, Facultad de Ingeniería, Cartagena de Indias. (2015)
3. Gaich,Facundo.: Técnicas mixtas de seguimiento y aprendizaje para tracking en secuencias de video. Facultad de Astronomía y Física, Universidad Nacional de Córdoba. (2015)
4. Riascos Segura, Johan Steve, Cardona Gallego, Yeison Alejandro.: Determinación de la cinemática de objetos móviles bajo condiciones controladas mediante imágenes empleando técnicas de flujo óptico. Universidad Tecnológica de Pereira. Facultad de Ingenierías. (2015).