JÓVENES EN LA CIENCIA
REVISTA DE DIVULGACIÓN CIENTÍFICA

Implementación de Algoritmos Usando la Plataforma CUDA Para el Procesamiento de Imágenes de Resonancia Magnética

((Banda Chávez, Juan Manuel (1)), (Serrano Rubio, Juan Pablo (2))

1 [Ingeniería en Sistemas Computacionales, Instituto Tecnológico Superior de Irapuato] | Dirección de correo electrónico: [jumbch@gmail.com]

2 [Maestría en Tecnologías de la Información, Instituto Tecnológico Superior de Irapuato] | Dirección de correo electrónico: [juserrano@itesi.edu.mx]

Resumen

En esta investigación se presenta un método multi-thresholding conocido como OTSU para identificar y segmentar los tumores cerebrales en imágenes de resonancia magnética. El algoritmo es implementado en la plataforma de procesamiento en paralelo CUDA la cual hace uso de tarjetas gráficas de la marca NVIDIA de 6ta generación, la librería OPENCV y el lenguaje de programación C++. El algoritmo OTSU segmenta la imagen en diferentes regiones mediante la maximización de la varianza entre los niveles de gris de la imagen. En los experimentos se utilizó la base de datos BRATS del Multimodal Brain Tumor Segmentation Challengue.

Abstract

In this investigation it's presented, a multi-thresholding method known as OTSU to identify and segment the brain tumors in images of magnetic resonance. The algorithm is implemented in the parallel processing platform CUDA, which makes us of graphics cards of the NVIDIA label of 6th generation, the OPENCV library, and the programming language C++. The OTSU algorithm segments the image in different regions by maximizing the variance between the gray levels of the image. In the experiments, the BRATS database was used, of the Multimodal Brain Tumor Segmentation Challenge.

Palabras Clave OTSU; Hilos; Computo en paralelo; NVIDIA; Procesamiento de Imágenes

INTRODUCCIÓN

El diseño de algoritmos computacionales aplicados al procesamiento de imágenes de resonancia magnética ayuda al tratamiento y diagnóstico de tumores cerebrales. La precisión en el diagnostico mejora la calidad de vida de los pacientes principalmente en la condición física, cognitiva y emocional, así como en su entorno familiar. El concepto de calidad de vida de los pacientes es un concepto multidimensional que comprende el aspecto social, físico, psicológico y está en función del correcto diagnóstico por parte del médico. La introducción de sistemas de reconocimiento tiene implicaciones económicas y sociales, ya que la correcta valoración del estado de los pacientes puede prevenir la muerte y discapacidades en los pacientes con tumores cerebrales.

MATERIALES Y MÉTODOS

Gracias al desarrollo del tratamiento digital de imágenes con métodos de extracción de información, mejora de la visualización o resaltado de rasgos de interés de las imágenes, se puede facilitar y mejorar el diagnóstico de los especialistas. Para poder empezar a trabajar con el tratamiento de imágenes se utilizó OpenCV ya que es una librería que se utiliza para el procesamiento digital de imágenes. Se creó una carpeta en el disco local reservada específicamente para la extracción de los archivos y componentes de la librería. Posteriormente se configuraron las variables de entorno del sistema para que la librería pudiera trabajar correctamente. Una vez configurado se creó un proyecto para comprobar si la librería funcionaba correctamente, el código y el programa se muestran a continuación en las imágenes 1 y 2. [1]

```
#include <opencv2/highgui/highgui.hpp>
#include <iostream>
using namespace cv;
using namespace std;
int main()
{
    string filename = "Fresa.jpg";
    Mat image = imread(filename, CV_LOAD_IMAGE_COLOR);
    if (!image.data)
        cout << "No se encontro la imagen" << std::endl;
    namedWindow("Display Window", WINDOW_AUTOSIZE);
    imshow("Display Window", image);
    waitKey(0);
    return 0;
}
```

Imagen 1. Código para mostrar una imagen en la pantalla.

Imagen 2. Programa en ejecución.

Se realizó un vínculo entre visual studio 2013 con la arquitectura CUDA ya que uno de los objetivos principales de este proyecto es que en el análisis de las imágenes de resonancia magnética sean analizadas lo más rápido posible por lo tanto se pensó trabajar con una tarjeta gráfica NVIDIA ya que esta cuenta con las características necesarias para realizar estas tareas permitiendo a la computadora trabajar en paralelo y obtener resultados más rápidamente. Una vez configurado se escribió el código para verificar que funcionara correctamente, la imagen 3 muestra parte del código de la aplicación y la imagen 4 los resultados. [3]

```
#include "stdafx.h"
#include <cuda.h>

#include <cuda_runtime.h>
#include "device_launch_parameters.h"
#include <stdio.h>
#include <iostream>
#define N 1000


__global__ void add()
{
    int col = blockIdx.x * blockDim.x + threadIdx.x;
    int fil = blockIdx.y * blockDim.y + threadIdx.y;
    int indice = col * 4 + fil;
    printf("Bloque = %d  Block_Size = %d Hilo = %d\n", blockIdx.x, blockDim.x, threadIdx.x);
}
```

Imagen 3. Código básico para conocer el funcionamiento de bloques, grids e hilos.

```
C:\Users\USER\Dropbox\Juan Manuel Banda Chavez\OpenCV\VEcCuda\Debug\VEcC
Bloque = 0  Block_Size = 3 Hilo = 0
Bloque = 0  Block_Size = 3 Hilo = 1
Bloque = 0  Block_Size = 3 Hilo = 2
Bloque = 0  Block_Size = 3 Hilo = 0
Bloque = 0  Block_Size = 3 Hilo = 1
Bloque = 0  Block_Size = 3 Hilo = 2
Bloque = 1  Block_Size = 3 Hilo = 0
Bloque = 1  Block_Size = 3 Hilo = 1
Bloque = 1  Block_Size = 3 Hilo = 2
Bloque = 1  Block_Size = 3 Hilo = 0
Bloque = 1  Block_Size = 3 Hilo = 1
Bloque = 1  Block_Size = 3 Hilo = 2
Bloque = 0  Block_Size = 3 Hilo = 0
Bloque = 0  Block_Size = 3 Hilo = 1
Bloque = 0  Block_Size = 3 Hilo = 2
Bloque = 0  Block_Size = 3 Hilo = 0
Bloque = 0  Block_Size = 3 Hilo = 1
Bloque = 0  Block_Size = 3 Hilo = 2
Bloque = 1  Block_Size = 3 Hilo = 0
Bloque = 1  Block_Size = 3 Hilo = 1
Bloque = 1  Block_Size = 3 Hilo = 2
Bloque = 1  Block_Size = 3 Hilo = 0
Bloque = 1  Block_Size = 3 Hilo = 1
Bloque = 1  Block_Size = 3 Hilo = 2
Presione una tecla para continuar . . .
```

Imagen 4. Programa en ejecución.

El proyecto fue desarrollado en un host con una tarjeta gráfica NVIDIA GTX630 de sexta generación la cual ejecutó múltiples algoritmos en paralelo. La programación paralela es el uso de varios procesadores trabajando en conjunto para dar solución a una tarea en común.

El algoritmo que se implementó para obtener un mejor resultado en la detección fue el algoritmo Otsu el cual fue desarrollado en 1979 por Nobuyuki Otsu cuya finalidad es separar objetos de una imagen del resto.[4] Con la ayuda de los métodos de valor umbral en las situaciones más sencillas se puede decidir qué píxeles conforman los objetos que buscamos y qué píxeles son sólo el entorno de estos objetos. La innovación sobre este algoritmo cae sobre computación en paralelo ya que se acelera el tiempo de procesamiento de las imágenes trabajando sobre la plataforma CUDA. Para poder convertir las imágenes de resonancia magnética a un formato compatible con OpenCV y CUDA se utilizó el software de análisis numérico MATLAB posteriormente se analizó la imagen que se muestra a continuación. (Ver Imagen 5)

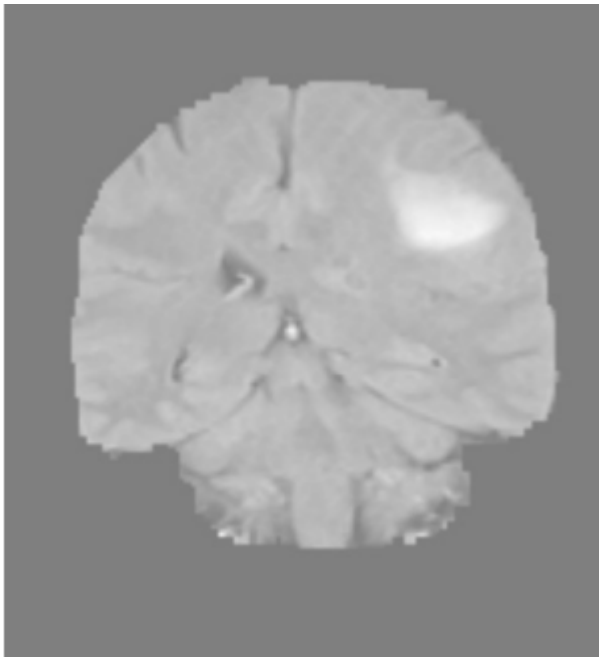


Imagen 5. Imagen obtenida a través de matlab

RESULTADOS Y DISCUSIÓN

El método de Otsu, llamado así en honor a Nobuyuki Otsu desarrollado en 1979, utiliza técnicas estadísticas, para resolver el problema. En concreto, se utiliza la varianza, que es una medida de la dispersión de valores (en este caso se trata de la dispersión de los niveles de gris). - Objetivo: calcular el valor umbral de forma que la dispersión dentro de cada clase sea lo más pequeña posible, pero al mismo tiempo la dispersión sea lo más alta posible entre clases diferentes. [4] El algoritmo se implementó en CUDA y en el lenguaje C++ de manera estructurada, la imagen analizada para ambos casos se muestra en la imagen 5. Finalmente, al umbralizar la imagen de un cerebro por el método de Otsu, se obtuvo como resultado una imagen, donde el color azul representa el fondo de la imagen, el naranja

el núcleo del tumor y el amarillo el tumor visible. (Ver Imagen 6)

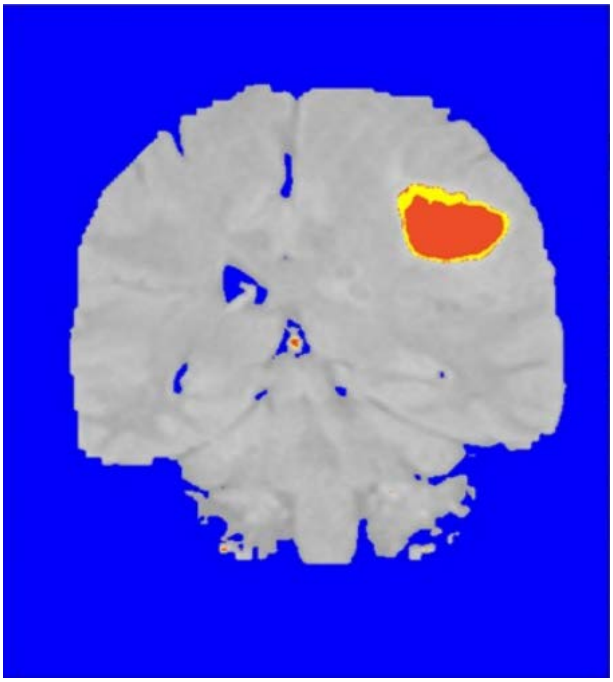
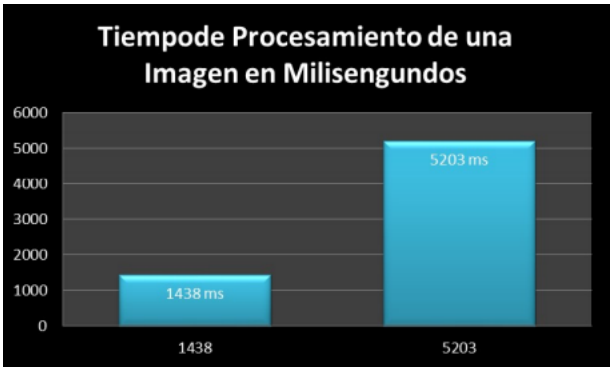


Imagen 6. Resultado del método umbral por algoritmo Otsu

CONCLUSIONES

De acuerdo a los resultados numéricos la implementación en paralelo del algoritmo OTSU ejecuta el algoritmo usando el 20% del tiempo que tarda la implementación en serie. Por otro lado, el algoritmo segmenta correctamente el tejido sano y el tejido con tumor de la imagen. A continuación, se presenta una tabla con la diferencia en tiempos de ejecución entre CUDA y C++ en serie. (Ver Tabla 1)

Tabla 1. Tabla comparativa entre CUDA y C++



AGRADECIMIENTOS

A mi asesor Juan Pablo Serrano Rubio por haberme apoyado a lo largo del desarrollo de este proyecto, al Instituto Tecnológico Superior de Irapuato quien proporciono los recursos tecnológicos para llevar a cabo esta investigación, a mi familia por siempre impulsarme a trabajar y salir adelante y a mi amigo Julio Cesar Rivera Espinosa, quien me asesoró en algunas ideas para la redacción de este artículo.

REFERENCIAS

[1] Gary, B., & Adrian, K. (2008). Learning OpenCV Computer Vision With the OpenCV Library. O'Reilly Media.
[2] Gonzalo, P., & Jesus, D. I. (2007). Vision por Computador: Imágenes Digitales y Aplicaciones. Alfaomega.
[3] Jason, S., & Edward, K. (2010). CUDA by Example. AddisonWesley.
[4] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.