Procesando el Reconocimiento de Múltiples Objetos en Tiempo Real Hacia la Implementación para el Reconocimiento de Actividades Humanas Concurrentes Utilizando GPU

Octavio I. Rentería-Vidales[a], Francisco E. Martínez Pérez[a], Juan C. Cuevas Tello[a], Omar Rodríguez González[a], Sandra E. Nava Muñoz[a]

[a]*Facultad de Ingeniería, Universidad Autónoma de San Luis Potosí*
Av. Dr. Manuel Nava No. 8, Zona Universitaria, San Luis Potosí, S.L.P., México, 78290
octavio.renteria@alumnos.uaslp.edu.mx, {eduardo.perez,cuevas,omarg,senavam}@uaslp.mx *2013 Published by*

$\mathcal{DIFU}_{100}ci@$ http://www2.uaz.edu.mx/web/www/publicaciones
Selection and peer-review under responsibility of the Organizing Committee of the CICOMP-2013, www.cicomp.org

Resumen

El presente artículo muestra los resultados preliminares de la implementación de una técnica de visión utilizando la unidad de procesamiento gráfica (GPU), aplicada al reconocimiento de actividades humanas entrelazadas y concurrentes. Se presenta la metodología de la implementación en pruebas realizadas en la unidad central de procesamiento (CPU) y utilizando GPU. Se muestra un ejemplo con el uso de cuatro filtros de correlación con el propósito de reconocer objetos en paralelo. Los resultados de los filtros de correlación son utilizados por el concepto Roaming Beat para dar un significado al comportamiento de los objetos dentro del desarrollo de actividades humanas. Se concluye que el uso de GPU es una opción factible en este tipo de sistemas.

Palabras clave: Procesamiento en paralelo, Reconocimiento de actividades, Reconocimiento de objetos.

1. Introducción

El reconocimiento de actividades humanas ha surgido como un problema de investigación en las últimas décadas. En el desarrollo de ambientes inteligentes, existen áreas tales como el cómputo ubico, el cómputo móvil, el cómputo consciente de contexto, entre otras, que requieren del reconocimiento de actividades y su entorno para responder a las necesidades de los usuarios [1]-[3].

En la actualidad una técnica para el reconocimiento de actividades es aquella basada en la utilización de

sensores vestibles tales como acelerómetros o identificadores por radiofrecuencias, entre otros [4]. Este tipo de tecnología es utilizada para el reconocimiento de actividades físicas tales como caminar, correr, sentarse, pararse, o bien actividades que tienen patrones de movimiento.

Otra de las técnicas que es utilizada en el reconocimiento de actividades es aquella basada en visión. Dentro de esta técnica se destaca la vigilancia visual inteligente que se refiere al proceso de monitorización visual automatizado, el cual involucra el análisis e interpretación del comportamiento de los objetos. Esto es realizado mediante el reconocimiento y seguimiento de objetos con el propósito de entender los eventos de la escena de una forma visual [5].

Ambas técnicas para el reconocimiento de actividades proporcionan patrones característicos de las señales reales, cuando una actividad se realiza o cuando se identifica una situación específica de un escenario. Esto ocurre para el reconocimiento de actividades físicas como para aquellas que se basan en las interacciones del uso de objetos [6]; por ejemplo, el reconocimiento de las actividades tales como el cepillarse los dientes, el prepararse un café en el hogar o el tomar la presión sanguínea en una residencia de cuidados.

Cualquiera que sea la actividad a reconocer, es necesario tomar en cuenta que las actividades no se realizan de forma aislada sino que se entrelazan o concurren con otras actividades. Este hecho genera una dificultad en el reconocimiento de las actividades debido a las diversas variantes en las que se pueden desarrollar. Es decir, las actividades concurrentes comienzan en un mismo instante por una o más personas y con uno o más objetos para interactuar, lo que lleva a realizar el reconocimiento de varios objetos así como el reconocimiento de su comportamiento en la interacción que se realiza. Lo mismo sucede con las actividades entrelazadas, en las cuales una actividad comienza a ejecutarse en un determinado tiempo, y antes de que termine esta actividad otra actividad comienza a ejecutarse. Este hecho lleva a realizar el reconocimiento de varios objetos y su comportamiento en un lapso de tiempo entrelazado.

El propósito de este artículo es mostrar los procedimientos y resultados preliminares en el reconocimiento de objetos en tiempo real, paralelizando los procesos, con el objetivo de obtener la inferencia de actividades entrelazadas y concurrentes. El proceso se muestra mediante una técnica de visión, que hace uso de filtros de correlación compuestos, y se presenta la descripción de la implementación de forma secuencial en una computadora y la implementación en paralelo de los filtros de correlación utilizando una Unidad Gráfica de Procesamiento (GPU por sus siglas en inglés *Graphics Processing Unit*).

El resto de este artículo está estructurado de la siguiente manera: en la sección 2 se muestra el trabajo relacionado, en la sección 3 se presenta la metodología de este artículo. En la sección 4 se muestran nuestros resultados preliminares y un ejemplo del desarrollo de actividades concurrentes utilizando nuestra implementación; por último se presentan nuestras conclusiones y las referencias.

2. Trabajo relacionado

El surgimiento de nuevos sistemas en ambientes reales ha sido posible gracias al avance de las nuevas tecnologías. Tal es el caso del uso de las unidades gráficas de procesamiento (GPU) que han propiciado la implementación del paralelismo como una alternativa para el futuro de la computación [7]. Algunos autores presentan un comparativo de la velocidad del CPU, OpenGL y CUDA utilizando una tarjeta de vídeo específica [7]. El propósito de nuestro trabajo no es el comenzar una discusión con respecto a las velocidades sino mostrar una implementación práctica y hacer uso del GPU como una alternativa en el reconocimiento de actividades entrelazadas y concurrentes.

Otros autores muestran una implementación en la cual hacen uso de filtrado de partículas, cuya característica es el costo computacional que se tiene [8]. Sin embargo, mencionan haber conseguido el desempeño en tiempo real necesario para su implementación. Otro ejemplo de implementación de la GPU es el desarrollo de una estrategia para el uso del GPU para la detección de objetos en movimiento [9].

Por otro lado, poco trabajo se ha reportado para el reconocimiento de actividades utilizando filtros de correlación y en menor proporción en el reconocimiento de actividades utilizando GPU. En este tipo de reconocimientos de actividades existen trabajos enfocados en el procesamiento de actividades entrelazadas o concurrentes entre los que destacan [10]-[12]. Éstos resaltan una técnica en particular denominada Patrones Emergentes (*Emerging Patterns - EP*). Esta técnica hace una interpretación de los datos de los sensores haciendo uso de una ventana deslizante en forma secuencial. Por lo que la captura y procesamiento se realiza de la misma manera.

Específicamente en sistemas para el reconocimiento de actividades se sugiere que los sistemas estén conformados por tres componentes tales como: i) un módulo de adquisición de información de bajo nivel; ii) un procesamiento y selección de características; y iii)

un módulo de clasificación que se encargue de utilizar las características para inferir las actividades [2]. Con este propósito, nuestro trabajo se basa en un concepto denominado Roaming Beat (RB) [13]. Este concepto hace uso de una metodología con la cual es posible obtener los tres componentes antes mencionados, y posteriormente llegar a reconocer las actividades entrelazadas y concurrentes, lo anterior será descrito en la siguiente sección.

3. Métodos

En esta sección se describen los pasos que fueron realizados en la implementación de los filtros de correlación, con el propósito de mostrar una alternativa para el reconocimiento de actividades entrelazadas y concurrentes.

3.1. Información Técnica

Los métodos implementados en este trabajo se realizaron en una computadora con procesador AMD FX-6100 con una frecuencia de reloj de 3.3/3.9 GHz; 6 MB en cache L2 y 8 MB en L3. El equipo contiene una tarjeta gráfica NVIDIA GeForce GTX 680, con una frecuencia 1006 /1058 MHz, 1536 núcleos CUDA; 2048 MB de memoria; una interfaz de memoria de 256-bit GDDR5 y un ancho de banda de 6 $Gbps$. La cámara utilizada fue una webcam Creative modelo VF0400 con una resolución de 640×480 pixeles.

Los métodos fueron realizados en C++ bajo la plataforma de Linux (Kernel 3.8.0) y con la ayuda de OpenCV, Armadillo y OpenCL[1] (para la implementación paralela).

La programación de GPUs se puede realizar utilizando CUDA u OpenCL. En este artículo se utiliza OpenCL porque permite su ejecución en cualquier tarjeta aceleradora gráfica como ATI Radeon (AMD), GeForce (NVIDIA) entre otras. Por otro lado, el código CUDA sólo se ejecuta en tarjetas de NVIDIA.

3.2. Implementación

El desarrollo de la implementación se realizó en dos etapas 1) Secuencial y 2) Paralela. Para ambas etapas se necesita el desarrollo de los filtros de correlación para el reconocimiento de objetos como se describe en [14]. En esta referencia se plantea el proceso de entrenamiento y el de las imágenes empleadas para considerar la traslación y rotación.

[1] http://www.khronos.org/opencl/

Además las dos etapas requirieron de la captura de la imagen que se obtiene del flujo de vídeo de la cámara web. Posteriormente se realiza un preprocesamiento a la imagen con el propósito de transformarla a escala de grises; recortarlas y adaptarlas al tamaño de los filtros (512×512) y aplicar un filtro pasa bajas mediante un blur Gaussiano con una matriz de 3×3. Este filtro fue utilizado con el propósito de suavizar la imagen de entrada, por lo que otro tipo de procesamiento tales como la detección de bordes queda fuera del alcance de este artículo, por el momento.

Un término que es utilizado en la programación de de la GPU es *kernel* [15]. Esta palabra se refiere a un identificador interno de la GPU que ejecuta un fragmento de código en paralelo mediante el uso de hilos (*Threads* o *work ítems*). Por lo que este término será utilizado en este trabajo para hacer referencia a la implementación en paralelo.

3.2.1. Implementación secuencial

La implementación realizada en esta etapa fue en forma secuencial. Es decir, se realizaron cuatro funciones que contenían lo siguiente:

1. Un filtro por objeto a reconocer realizado con base en el algoritmo presentado en [14].
2. Tomar una imagen de entrada de la cámara de vídeo.
3. Aplicar la Transformada de Fourier (TF) a la imagen de entrada.
4. Aplicar la Ley K a la imagen de entrada. que es un filtro compuesto SDF no lineal [17]-[18] que tiene tolerancia a la deformación (rotación y traslación) de algunos objetos y un buen desempeño en la presencia de diferentes tipos de ruido, todo esto en el reconocimiento de objetos.
5. Realizar la correlación del filtro con la imagen de entrada.
6. Aplicar al resultado de la correlación la TF inversa

Dentro de un ciclo se ejecutaban las cuatro funciones. En el CPU cada pixel es procesado secuencialmente hasta completar la imagen.

3.2.2. Implementación en paralelo

Para la paralelización de procesos se empleó OpenCL para la programación de la GPU. No todos los pasos fueron paralelizados, sólo aquellos que eran críticos y consumían mucho tiempo como los antes mencionados en la implementación secuencial.

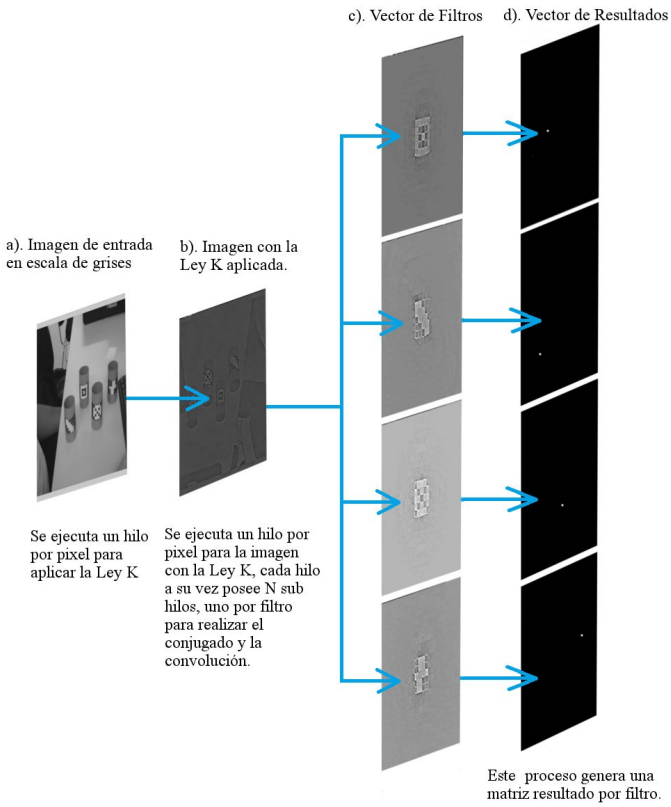

Figura 1. Representación de cuatro procesos paralelos implementados en GPU

El desarrollo consistió en implementar las siguientes funciones:

1. Se implementó un kernel para ejecutar la TF. En este módulo, fue necesario realizar las ecuaciones de la TF para implementarlas en OpenCL. Este procesamiento permitió realizar el código portable; es decir, poder implementar el mismo código en las tarjetas de vídeo AMD® o INTEL®. Esta implementación se basó en el algoritmo de la TF *Cooley-Tukey* [16]. Es importante mencionar que existen algoritmos de la TF. Sin embargo, hasta lo que se ha investigado, estos algoritmos están diseñados para ser implementados en CPU y no en GPU para el procesamiento en paralelo.
2. Un kernel para la función discriminante sintética llamada Ley K;
3. Un kernel para realizar la correlación.
4. Un kernel para implementar la transformada inversa de Fourier

Estos métodos son muy críticos por el procesamiento computacional que se requiere. Las imágenes fueron vectorizadas para aplicar la TF.

Una vez desarrollados los kernels se realizaron los siguientes pasos: Primeramente se copia la imagen adquirida de la cámara en la memoria de vídeo de la GPU, esto con el propósito de evitar costosos tiempos de transferencia de datos entre el CPU y la GPU. Una vez en ella se ejecuta el kernel de cada función comenzando con aplicar la TF a la imagen de entrada. Posteriormente se transfiere el resultado al siguiente kernel que es la Ley K, en la cual se le aplica a la misma imagen (Figura 1 b)). Después, se ejecuta la correlación del filtro del objeto a reconocer con los resultados del kernel anterior. En este proceso, aunque se plantea de forma secuencial, en realidad se ejecuta la correlación de cada filtro con la imagen en forma paralela. En las Figuras 1 b) y 1 c) se muestra el proceso, en donde la imagen con la Ley K aplicada, se correlaciona con cada uno de los filtros en forma paralela.

Una vez realizada la correlación, se obtiene un vector por cada filtro con los resultados como lo muestra la Figura 1 d). A estos resultados se les aplica la transformada inversa de Fourier en forma paralela. La implementación del procesamiento en paralelo es realizado internamente por la tarjeta de vídeo.

Finalmente, el vector resultante de cada filtro es convertido a una matriz cuadrada, posteriormente se realiza un intercambio de los cuadrantes de la imagen (ifftshift) para poderla visualizarla.

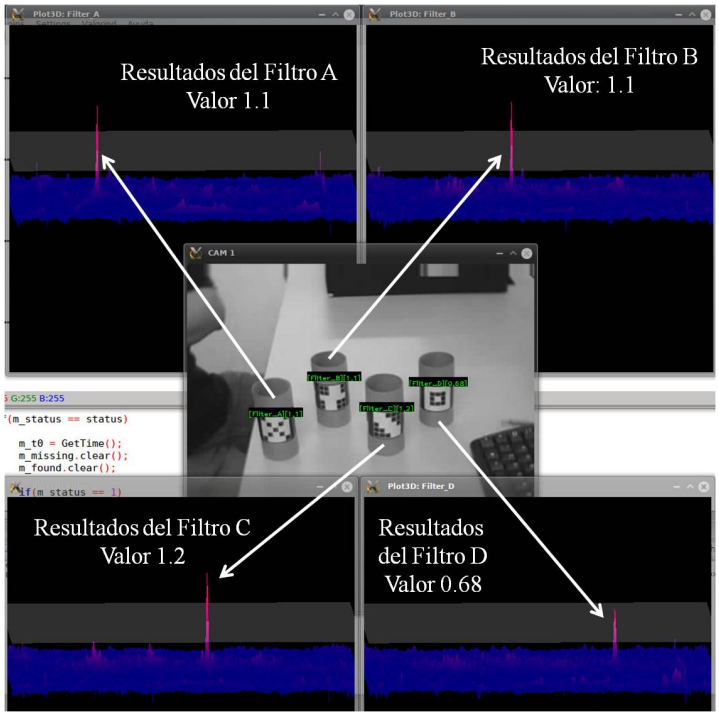

Figura 2. Resultados de cuatro filtros de correlación implementados en la GPU

La Figura 2 muestra los resultados de correlación realizados con la GPU en tiempo real. En esta figura, se pueden observar cuatro objetos con diferentes etiquetas. Cada una de las ventanas ilustra el reconocimiento de su objeto, mostrando un pico muy alto, el cual indica la presencia del objeto en la escena. El valor alto representa las coordenadas (x, y), en donde se localiza el objeto en la imagen entrante. Filtro A con un valor de reconocimiento de 1.1; filtro B con un valor de reconocimiento de 1.1; filtro C con un valor de reconocimiento de 1.2 y filtro D con un valor de reconocimiento de 0.68. Se puede observar que cada filtro reconoce a su objeto en específico y que la presencia de otras etiquetas no produce un valor significativo que pueda causar confusión. Esto se debe al umbral que se considera, con el cual se clasifica como objeto encontrado si excede el umbral, de lo contrario se descarta.

En la GPU, al implementar el kernel, todos los pixeles adquieren un hilo independiente, por lo que se ejecutan simultáneamente dependiendo de la implementación y de las capacidades internas de la tarjeta utilizada. La forma en que lo realiza la tarjeta produce una disminución del tiempo de procesamiento. Por ejemplo, considere un vector de 16 elementos, para el cual se tiene que definir un número de hilos a ejecutar. Cada hilo resolverá 4 elementos del vector, por lo cual es necesario dividir el vector entre 4. En este caso se utilizarán 4 hilos. Por lo que cada uno de los hilos se encarga de ejecutar un proceso a 4 elementos de forma independiente. De esta forma se optimiza internamente el procesamiento. Para mayor detalle de la programación de OpenCL para GPU consultar [19].

3.3. Procesamiento de la información

Una vez realizada la correlación de las imágenes en la secuencia de vídeo, el siguiente paso es realizar un procesamiento y la selección de características. Para la interpretación de la selección de las características se utilizó el concepto Roaming Beat el cual se define como:

> "La habilidad de un artefacto para dar una marca de tiempo (hora y fecha) para cambiar de un estado sin movimiento a uno con movimiento, de una localización base a otra posición errante [20]"

Este concepto hace uso de una metodología con la cual es posible inferir actividades entrelazadas y concurrentes. El concepto se basa en la obtención de un comportamiento a partir de la manipulación de objetos/artefactos en el desarrollo de una actividad.

Para lograr una selección de objetos se utilizan los resultados de la correlación de cada una de las imágenes de la secuencia de vídeo. Una de las ventajas de haber utilizado este tipo de filtros de correlación es que es posible reconocer varios objetos de la misma clase con un

solo filtro. Además de que es posible considerar el reconocimiento de objetos tomando en cuenta la traslación y rotación dentro de un mismo filtro. Al implementarse los filtros en paralelo utilizando GPU, es posible realizar el análisis del comportamiento de cada uno de los objetos de forma independiente, y que a su vez se puedan coordinar con los demás objetos participantes en una actividad.

Por lo que cada uno de los resultados de la correlación se compara con un umbral. Es decir, se obtienen los picos de correlación de cada una de las imágenes que son los valores más grandes como resultados del hallazgo de uno o más objetos en la imagen de la escena. Si los valores exceden el umbral se considera el reconocimiento del objeto y se interpreta como una señal que forma parte del comportamiento en el desarrollo de la actividad. Si los valores de correlación resultantes no exceden el umbral, se interpreta como una ausencia del objeto en la escena. En esta etapa se obtienen los valores máximos haciendo un recorrido de los resultados de los filtros de correlación en forma secuencial. De esta forma se discretizan los resultados de los filtros de correlación y se obtiene la representación del concepto de RB de cada uno de los filtros implementados.

Por ejemplo, en la Figura 3 considere el filtro A de la imagen. Este filtro comienza con el reconocimiento del artefacto, es decir, los resultados de la correlación con este filtro muestran un valor que se encontraba por encima del umbral. Por consiguiente, el valor es representado con un 1 desde el inicio de la secuencia de vídeo. Posteriormente en el índice 146 ocurre un cambio de 1 a 0. Es decir el artefacto sale de la escena y no es reconocido por el filtro. De este forma se discretizan los resultado de correlación y se va formando un comportamiento hasta cumplir las reglas para inferir la actividad como se menciona en [13]. En este caso, el comportamiento de este filtro debe de cumplir al menos 5 cambios de estado de 0 a 1 o viceversa. Este filtro obtuvo 6 cambios de estado (en los índices 284, 344, 410, 457, 558 y 661 de la Figura 3) y pudo inferirse la actividad.

4. Resultados

Se realizaron pruebas experimentales en laboratorio, en el cual se instaló una cámara web para realizar el reconocimiento de artefactos. La cámara genera 18 cuadros por segundo, bajo la plataforma Linux, a los cuales se les realizó el procesamiento.

En este trabajo se utilizaron etiquetas para realizar el reconocimiento de objetos como se muestra en la Figura 2. Además, es posible realizar el reconocimiento de objetos utilizando las características físicas de los objetos. Los objetos utilizados en estas pruebas corresponden a un ambiente de cuidados de salud. Cada etiqueta representa un artefacto. El filtro A de la Figura 3 representa a una solución fisiológica relacionada con la actividad de curación; el filtro B representa el uso del papel higiénico relacionado con la actividad de higiene; el filtro C representa el objeto para tomar la presión arterial relacionada con la actividad para conocer la presión; y el último filtro representa al objeto para medir la glucosa relacionada con la actividad para medir los niveles de azúcar en la sangre. Aunque algunas de estas actividades están relacionadas con dos o más objetos, en esta demostración se asume que estas actividades se pueden inferir reconociendo sólo un objeto.

Tabla 1. Tiempos promedios en milisegundos

	Fourier	Ley K	Correlación	Fourier Inversa
CPU	9.21	73.83	22.66	9.26
GPU	0.35	0.08	0.04	0.40

Como se mencionó anteriormente, se realizaron cuatro filtros de correlación que corresponden a cuatro objetos como se muestran en la Figura 3 (filtros A, B, C y D). Los filtros no consideraron imágenes para discriminar otro tipo de objetos. Estos filtros consideraron tres imágenes de referencia con las cuales se considera la traslación. Para todos los filtros se manejó un umbral de 0.65, por lo que todo valor por encima de este umbral es 1, y 0 para valor por debajo de umbral.

La Tabla 1 muestra los tiempos obtenidos por un filtro. Se realizaron 100 iteraciones de este filtro con 100 imágenes entrantes de la cámara de vídeo. Los filtros y las imágenes fueron de tamaño 512×512. Las imágenes entrantes de la cámara de vídeo se recortaron a 512 y se rellenaron con ceros para cumplir la especificación de los filtros (512), quedando las imágenes centradas. En la Tabla 1 se puede observar mejores resultados con el uso de la GPU. Esto se debe a que en esta implementación los pixeles de cada imagen son procesados por un hilo independiente, disminuyendo los tiempos de procesamiento.

La Figura 2 muestra los resultados de la implementación en paralelo mediante la GPU. Son cuatro filtros, de los cuales tres de ellos (filtros A, B, C) muestran un resultado de correlación por encima de 1.0. Se observa como el pico de los tres filtros y el filtro D con un resultado de 0.68. Todos se encuentran por encima del umbral, lo cual es requisito para discretizar la información del comportamiento de los objetos. Un detalle importante en estos resultados es que al momento de ir insertando cada uno de los objetos en el escenario, el filtro del objeto entrante inmediatamente mostraba su

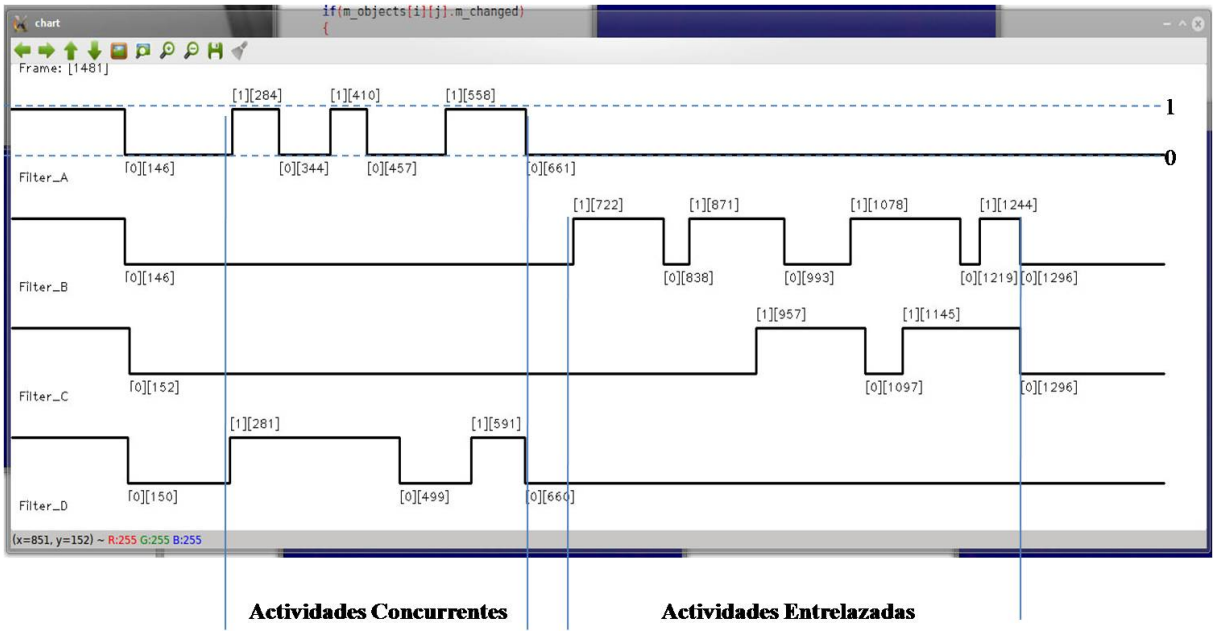

Figura 3. Ejemplo de la ejecucion de actividades entrelazadas y concurrentes

pico correspondiente del reconocimiento. Por otro lado, cuando se insertaban los objetos que no correspondían al filtro, se mostraba un resultado de correlación en el rango de 0.20 y 0.30, suficiente para descartar esta información. Siendo con esto suficiente para utilizar el concepto RB e implementar la inferencia de las actividades entrelazadas y concurrentes, lo cual se describe a continuación.

4.1. Seguimiento del reconocimiento de objetos en paralelo

Este seguimiento del reconocimiento corresponde a la interpretación del comportamiento de los artefactos en el desarrollo de una actividad. Para lo cual se realizaron dos ejemplos de la ejecución de actividades concurrentes y entrelazadas. El primer ejemplo corresponde a la ejecución de dos actividades que están relacionadas con un solo objeto (Filtro A y D de la Figura 3. En este ejemplo ambos comportamientos comienzan y terminan en los mismos tiempos (281, 284 y 660 y 661 respectivamente). El segundo ejemplo corresponde a la ejecución de dos actividades pero de forma entrelazada (Filtros B y C de la Figura 3). La primera de ellas comienza en un tiempo (índice 722 del filtro B) y posteriormente se integra la segunda actividad (índice 957 del filtro C). En este ejemplo, ambas actividades terminan en el mismo tiempo (índice 1296 de los filtros B y C).

En la Figura 3, además del comportamiento de los artefactos, en cada cambio de estado de 0 a 1 ó beat, se muestran los índices de la secuencia de vídeo de la cámara web. Con este índice se puede observar una diferencia mínima entre un reconocimiento y otro de los dos artefactos. Por ejemplo, en la actividad concurrente, el primer objeto (filtro D en la Figura 3) se reconoce en la imagen 281 mientras que el segundo objeto (filtro A de la Figura 3) se reconoce en la imagen 284. En la finalización de estas actividades el primer objeto (filtro D de la Figura 3) desaparece de escena en la imagen 660 y el segundo en la imagen 661.

Con estos resultados se corrobora la factibilidad de poder implementar las actividades entrelazadas y concurrentes a partir de utilizar el concepto RB y el utilizar un procesamiento mediante GPU. Esto, con base a que las actividades que realiza el humano son impredecibles y que es necesario realizar un procesamiento en paralelo para estar a la expectativa de todos los artefactos que sean necesarios.

5. Conclusiones y discusión

Se concluye que el uso de la GPU es una opción factible para la implementación de sistemas para el reconocimiento de actividades entrelazadas y concurrentes. Adicionalmente, es necesario tener bien definido las implicaciones para el reconocimiento de actividades. En este sentido, el utilizar el concepto RB fue de gran utilidad al marcar la pauta por el seguimiento de un comportamiento. Esto a partir del reconocimiento de los objetos con los cuales interactúa un humano en el desarrollo de una actividad.

Se utilizó una técnica de visión relacionada con filtros

de correlación compuestos. En el pasado se mostraba esta técnica para el seguimiento y detección de objetos, como costosa computacionalmente [21]. Sin embargo, con el uso de GPU, u otro tipo de tecnología, esta técnica puede llegar a ser implementada hasta en dispositivos embebidos, permitiendo el múltiple reconocimiento de objetos en segmentos de video en tiempo real.

El propósito de este artículo fue mostrar los resultados preliminares para la implementación de sistemas en el reconocimiento de actividades considerando las diversas formas que tiene el ser humano para realizarlas. Por lo que sabemos las implicaciones con las que nos enfrentamos y que los tiempos aquí mostrados pueden ser subjetivos. Nosotros consideramos estos resultados como pertinentes para continuar con nuestra investigación.

Este trabajo presenta dos de los tres componentes necesarios para un sistema de reconocimiento de actividades [2]. Por lo que nuestro trabajo futuro se enfocaría en la creación de un módulo de clasificación que se encargue de utilizar las características para inferir las actividades de una forma proactiva, inteligente y automática. Además de trabajar en un algoritmo secuencial equivalente a la paralelización mostrada en este articulo que permita medir la complejidad computacional. Adicionalmente, dirigir los esfuerzos en una optimización de los filtros de correlación compuestos para el reconocimiento de objetos similar al que se presenta en [9] utilizando GPU. Otro de los tópicos importantes en el uso de técnicas de visión es el uso de la oclusión y otras distorsiones, lo cual se plantea como trabajo futuro mediante la implementación de otro tipo de tecnología.

6. Agradecimiento

Este trabajo fue apoyado para su realización y presentación en congreso por el proyecto PROMEP del segundo autor con el número de oficio de la carta de liberación PROMEP/103.5/13/6575.

Referencias

[1] M. Weiser, “The computer for the 21st century,” *Scientific American*, Vol. 265, No. 3, pp. 94-104, 1991.

[2] T. Choudhury *et al*, “The Mobile Sensing Platform: An Embedded Activity Recognition System,” *Pervasive Computing*, pp. 32-41, 2008.

[3] A. K. Dey and G. D. Abowd, “Towards a better understanding of context and context-awareness,” *CHI 2000 workshop on the what, who, where, when, and how of context-awareness*, 2000, Vol. 4, pp. 1-6.

[4] L. Chen, J. Hoey, C. Nugent, D. Cook, and Z. Yu, “Sensor-based activity recognition,” *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 42, No. 6, pp. 790-808, 2012.

[5] I. S. Kim, H. S. Choi, K. M. Yi, J. Y. Choi, and S. G. Kong, “Intelligent visual surveillance - A survey,” *International Journal of Control, Automation and Systems*, Vol. 8, No. 5, pp. 926-939, Oct. 2010.

[6] M. Philipose *et al*, “Inferring activities from interactions with objects,” *IEEE Pervasive Computing*, Vol. 3, pp. 50-57, 2004.

[7] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, “GPU Computing,” *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879-899, May 2008.

[8] W. Limprasert, A. Wallace, and G. Michaelson, “Real-Time People Tracking in a Camera Network,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, Vol. 3, No. 2, pp. 263-271, Jun. 2013.

[9] D. Berjón, C. Cuevas, F. Morán, and N. García, “GPU-based Implementation of an Optimized Nonparametric Background Modeling for Real-time Moving Object Detection,” *IEEE Transactions on Consumer Electronics*, Vol. 59, No. 2, pp. 361-369, 2013.

[10] E. Kim, S. Helal, and D. Cook, “Human activity recognition and pattern discovery,” *Pervasive Computing, IEEE*, Nol. 9, No. 1, pp. 48-53, 2010.

[11] T. Gu, Z. Wu, X. Tao, H. Keng Pung, and J. Lu, “epSICAR: An Emerging Patterns based approach to sequential, interleaved and Concurrent Activity Recognition,” *Pervasive Computing and Communications, 2009. PerCom 2009*. IEEE International Conference on, 2009, pp. 1-9.

[12] T. Gu, L. Wang, Z. Wu, and X. Tao, “A pattern mining approach to sensor-based human activity recognition,” *IEEE Transactions on Knowledge and Data Engineering*, Vol. 23, No. 9, pp. 1359-1372, 2011.

[13] F. E. Martínez-Pérez, J. Á. González-Fraga, J. C. Cuevas-Tello, and M. D. Rodríguez, “Activity Inference for Ambient Intelligence Through Handling Artifacts in a Healthcare Environment,” *Sensors*, Vol. 12, No. 1, pp. 1072-1099, Jan. 2012.

[14] F. E. Martinez-Perez, J. González-Fraga, and M. Tentori, “Automatic activity estimation based on object behaviour signature,” *Proceedings of SPIE, 2010*, Vol. 7798, No. 1, p. 77980E.

[15] *OpenCL Programming Guide for the CUDA Architecture*, Ver. 3.2. 2009, p. 61.

[16] E. Ziegel, W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Vol. 29, No. 4. 1987, p. 501.

[17] B. Javidi and D. Painchaud, “Distortion-invariant pattern recognition with Fourier-plane nonlinear filter,” *Appl. Optics*, Vol. 35, No. 2, pp. 318-331, 1996.

[18] B. Javidi, W. Wang, and G. Zhang, “Composite Fourier-plane nonlinear filter for distortion-invariant pattern recognition,” *Optical Engineering*, Vol. 36, No. 10, 1997.

[19] A. Munshi, B. R. Gaster, T. G. Mattson, J. Fung, and D. Ginsburg, *OpenCL programming guide*. Addison-Wesley, 2011, p. 603.

[20] F. E. Martinez-Perez, J. A. Gonzalez-Fraga, and M. Tentori, “Artifacts’Roaming Beats Recognition for Estimating Care Activities in a Nursing Home,” *4th International Conference on Pervasive Computing Technologies for Healthcare 2010*, 2010.

[21] W. Hu, T. Tan, L. Wang, and S. Maybank, “A survey on visual surveillance of object motion and behaviors,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 34, No. 3, pp. 334-352, 2004.