Algorithm to Expand Storage Capacity in Video Security Systems Using Dynamic Filters (Software DifferentFiles)

Mario Martínez, Doctorado en Investigación e Innovación Educativa, Luis Barrera, Ingeniería Mecatrónica, Rene Osorio, Doctorado en Electrónica de Potencia, Jorge Brizuela, Doctorado en Ciencias en Ingeniería Electrónica
Universidad de Guadalajara, México, mariom@valles.udg.mx, rene.osorio@profesores.valles.udg.mx
barrier2210@gmail.com, jorge.brizuela@valles.udg.mx.

***Abstract*– This study aims to propose a novel method for the identification of repeated images stored in a security camera system using an algorithm that discriminates image weight by sorting them in a queue based on their last modification date. Dynamic filters are applied for identifying these differences, which contribute to the improvement of event search accuracy by only exposing the desired events and freeing space in system storage devices by discarding repeated images.**

***Keywords*–Computing, Software, Security, Storage Saving, Dynamic Filters, Fog Computing, Programming.**

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2017.1.1.250
ISBN: 978-0-9993443-0-9
ISSN: 2414-6390

Algoritmo para ampliar la capacidad de almacenamiento en sistemas de video seguridad aplicando filtros dinámicos (Software DifferentFiles)

Mario Martínez, Doctorado en Investigación e Innovación Educativa, Luis Barrera, Ingeniería Mecatrónica, Rene Osorio, Doctorado en Electrónica de Potencia, Jorge Brizuela, Doctorado en Ciencias en Ingeniería Electrónica
Universidad de Guadalajara, México, mariom@valles.udg.mx, rene.osorio@profesores.valles.udg.mx
barrier2210@gmail.com, jorge.brizuela@valles.udg.mx.

Resumen– La presente invención expone un método novedoso en la identificación de imágenes repetidas dentro de un sistema de cámaras de seguridad, proponiendo un algoritmo que discrimina el peso de las imágenes mediante un orden dependiendo de la fecha de modificación en tipo cola. Dentro de la identificación de dichas diferencias se aplican filtros dinámicos, los cuales contribuyen a mejorar la precisión de búsqueda de distintos eventos exponiendo los deseados, ayudando a liberar espacio descartando las imágenes repetidas logrando menor saturación en los dispositivos de almacenamiento del sistema.

Palabras Clave – Computación, Software, Seguridad, ahorro de almacenamiento, Filtros dinámicos, Fog Computing, Programación.

I. Introducción

La necesidad de instalar sistemas de seguridad con cámaras es cada vez más notorio ante una tecnología más accesible para mini negocios o lugares privados, lo que lleva a una amplia gama de sistemas que exponen distintos métodos de captura, ya sea por imágenes, video o cámaras sensibles al movimiento. Tales tecnologías incorporan un conjunto de herramientas informáticas y computacionales abriendo un panorama para el diseño y optimización de los sistemas que coadyuva con diversos métodos de identificación de rostros, movimiento, procesos industriales para la detección de colores o formas entre otros. [1]

El desarrollo de un software para el desempeño de sistemas de seguridad es crucial para el mejor desempeño del hardware. Los grandes consumidores de sistemas de vigilancia suelen tener operadores humanos que tienen la tarea de monitorear y analizar el panorama en tiempo real, el principal problema recae en que los operadores tienen a cargo diversas cámaras y pueden perder detalles a la par que en las revisiones invierten mucho tiempo. [2]

Las imágenes y videos captados tienden a albergar gran cantidad de espacios en las unidades de almacenamiento de los equipos de cómputo, definidos para estas tareas, así también se destaca la gran cantidad de imágenes sin contenido relevante como en los ejemplos de lugares de comercio cerrados donde no se encuentra alguna clase de movimiento en el transcurso donde el lugar permanece cerrado, sin embargo esta información usa espacio en el disco, llenándolo de eventos insignificantes y repetitivos.

La solución que se propone, es una discriminación con base al peso en bytes de la imagen en comparación a la siguiente en una línea ordenada por tiempo de modificación.

El programa está desarrollado en lenguaje C/C++ en conjunto con comandos CMD (*Command Prompt*) utilizando el compilador Dev C++, se optó por usar dichas herramientas y lenguajes para poder trabajar a un nivel de procesamiento rápido y con menos ciclos máquina, ya que el lenguaje que se usa en conjunto con el CMD facilitando los algoritmos para Windows. El rendimiento de un equipo de cómputo también marca gran diferencia en cuanto al tiempo de procesamiento y discriminación de las imágenes repetidas.

La solución propuesta consulta la información de cada archivo sin necesidad de abrir ninguna imagen, gracias a ello el trabajo es más rápido en el procesamiento de la información y al realizar las tareas para discriminar la información repetida, dentro los parámetros o filtros establecidos en la aplicación.

El Software es utilizado con un conjunto de fotografías recopiladas de un sistema de seguridad. El desarrollo es flexible, respecto a que puede procesar información de distintos tipos de formato de imágenes, tamaño y antigüedad de captura, por ello se puede escoger diferentes formatos de imágenes como lo son jpg, png, tratando a los mismos como archivos, consultando sus propiedades y haciendo una limpieza de carpeta principal migrándolas a otras áreas dependiendo de los filtros y sus características e información almacenada.

II. Antecedentes

El *Fog Computing*, permite una nueva generación de conexiones a través de aplicaciones y servicios extendiendo el paradigma y utilidad global del *cloud computing*. Las características que definen el *fog computing* son el conocimiento de baja latencia, gran número de nodos, función predominante del acceso inalámbrico, presencia fuerte de aplicaciones *streaming* y tiempo real entre otras. [3]

Las características anteriores que incorpora el *fog computing* son adecuadas para casos específicos como servicios y aplicaciones que ayudan dentro del internet de las cosas (*IoT*). Permitiendo de esta forma que a través del *fog*

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2017.1.1.250
ISBN: 978-0-9993443-0-9
ISSN: 2414-6390

computing se discrimine la información basura o repetida presente en nuestros servicios, con un pre-procesamiento para aliviar la gran cantidad de espacio de almacenamiento que se requeriría si no se realiza esta discriminación.

Los procesos ya existentes del *cloud computing* nos brindan diversas utilidades de optimización de espacios en dispositivos de almacenamiento, por el hecho de que se tiene gran cantidad de archivos creados y almacenados, lo cual se vuelve una necesidad para implementar programas que ayuden a mejorar el desempeño y ahorro de espacio, energía, tiempo y ciclos de microprocesador para contribuir a la economía de infraestructura en empresas públicas y privadas ligadas al Internet de las Cosas (IoT).

Existen algunas soluciones para no desperdiciar espacio de almacenamiento en información repetida en diferentes unidades de tiempo, por ejemplo: sistemas que solo almacenan video o imágenes cuando existe movimiento, es buena esta propuesta, pero la propuesta pierde información en cuanto al estado perpetuado a lo largo del periodo en el cual no se tuvo movimiento, lo cual tiende a ser parámetros muy austeros para la detección de anomalías mínimas como lo son sombras u objetos dentro de una transición además de la escasa versatilidad de sus filtros que manejan este tipo de sistemas automáticos.

Existen otros tipos de sistemas de vigilancia, que son los empleados para la observación de vida silvestre, la necesidad de obtener fotografías por presencia de anomalías o cambios en la escena obtener por medio de sensores, en este caso de la captura de imágenes digitales y video de 35mm con cámaras nocturnas con ayuda de un sistema con sensores infrarrojos que monitorea la escena para la posterior identificación de especímenes no-objetivo u objetivo en entorno silvestre o urbano, la deficiencia que pasa en estos casos es que los que utilizan sensores de movimiento como sensores infrarrojos, pero el problema recae en la pérdida de información en el lapso de tiempo anterior y posterior al suceso que no llega a ser captado por un sensor de baja sensibilidad y con poco rango de trabajo, con ello la vigilancia anterior y posterior se pierde abruptamente. [4]

III. Planteamiento

El sistema de monitoreo presentado busca la identificación de imágenes repetidas en una secuencia de archivos jpg o png con el fin de limpiar la unidad de almacenamiento de los sistemas de cámaras de vigilancia sin necesidad de procesar la imagen en forma de matriz o con un reconocimiento de patrones, esto para hacer óptimo el rendimiento del sistema en un equipo de cómputo convencional ya sea doméstico o de baja capacidad de procesamiento. Se determinan las imágenes repetidas o iguales dependiendo del filtro y tolerancia que se desee, las imágenes que su peso sea aceptable dentro el rango mayor o menor de peso proporcional con respecto a los filtros el sistema la tomará como igual a la siguiente previamente consultada para hacer la comparación.

Las imágenes definidas como diferentes, son aquellas que tienen un peso desequilibrado a la tolerancia anteriormente señalada esto se debe al tipo de compresión y codificación de estos archivos tipo imagen, donde la resolución, saturación de color o cambios en escena como luces o colores alteran significativamente el peso de la imagen alojada en un dispositivo de almacenamiento.[5]

En los métodos de monitoreo con sistemas de vigilancia por medio de fotos, imágenes, video, diagramas, ilustraciones, etc. es importante que se pueda determinar si hay una anomalía en el sistema comparando los datos o archivos actuales a los anteriores para posteriormente realizar una acción respectiva. [6]

Actualmente la sociedad requiere registrar eventos a través de circuitos de seguridad en espacios públicos, oficinas, almacenes, empresas, entradas, salidas, hogar, campos militares, entre otros. Este tipo de circuitos de vigilancia dependen de dispositivos de almacenamiento y la capacidad de almacenamiento es finita. [7]

Si contamos con una unidad de almacenamiento con una capacidad o memoria máxima deseada para alojar las imágenes tomado en bytes como la variable "t" y un circuito de seguridad que almacena cada segundo y un conjunto de imágenes de distintos tamaños siendo un total de "x" imágenes y "T" el peso total de las "x" imágenes. Logrando una estimación haciendo la ecuación (1).

"x" también representa el tiempo total registrado en el dispositivo de almacenamiento en la unidad de tiempo-segundo, una vez que el espacio es saturado en el dispositivo de almacenamiento, la información se empieza a reescribir y por lo tanto se pierde información.

TABLA I
DEFINICIÓN DE LAS VARIABLES PARA ECUACIÓN (1)

Variable	Valor
T	Variable para definir el espacio utilizado por las imágenes seleccionadas en una unidad de almacenamiento.
x	Número total de imágenes leídas en el directorio del programa para procesamiento y comparación.
t	Tiempo estimado para que se sature la memoria deseada.
i_n	Donde i_n, es el nombre de la imagen
S_n	Propiedad de la imagen a evaluar

$$f(T,x) = \frac{T}{\left(\frac{\sum_{i=1}^{x} p_n}{x} \right)} = t \qquad (1)$$

El mismo espacio se requiere para almacenar 12 horas de día que 12 horas de noche en un dispositivo de almacenamiento de un circuito cerrado de video o imágenes, o 24 horas en una oficina en periodo laboral que en periodo de vacaciones o en el hogar, aun cuando existen amplias diferencias en cuanto a actividad, tanto de día como de noche o en periodo laboral y de asueto.

III. Desarrollo

La solución propuesta es representar con un arreglo las imágenes repetidas (1) con su respectiva magnitud que se presentaron en un lapso de tiempo, esto representa casos de la misma imagen almacenada con la misma magnitud (n veces) en distinto lapso de tiempo hasta alcanzar una saturación y uso de recursos innecesarios dentro el sistema de vigilancia. Los eventos que se presentan de forma diferenciada continuamente dentro de los sistemas de seguridad, que posean eventos relevantes tales como (3) es un caso distinto puesto que pasarán a almacenarse sin problemas al arreglo. Por otro lado se apreciaría un arreglo, que tiene sólo las imágenes diferentes al resto, lo cual se expresaría de la siguiente forma (3) reemplazando el arreglo original de imágenes combinadas repetidas y distintas para poder adquirir un arreglo único de imágenes diferentes.[8]

Tabla II
Valores de las variables para ecuaciones (2, 3, 4)

Variable	Valor
D	A1 arreglo dentro del vector para analizar.

$$D_1 = [(i_1, P_1) = (i_2, P_2) = (i_3, P_3) = (i_4, P_4)...(i_n, P_n)] \qquad (2)$$

$$D_2 = [(i_4, P_4) = (i_5, P_5) \neq (i_6, P_6) \neq (i_7, P_7)...(i_n, P_n)] \qquad (3)$$

$$D_3 = [(i_5, P_5) \neq (i_6, P_6) \neq (i_7, P_7)...(i_n, P_n)] \qquad (4)$$

El orden de la estructura lineal de imágenes es dependiente de la fecha de modificación, haciendo un orden tipo cola o también conocido como FIFO (*First In First Out* ó en español primero dentro, primero fuera), como se muestra en la Fig. 1, señalando también la orientación que sigue el procesamiento de imágenes dentro de la estructura, añadiendo de lado izquierdo imágenes nuevas y en la derecha con una transición hacia la izquierda expresando la dirección del procesamiento en tipo cola.

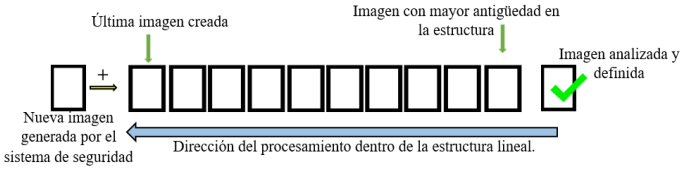

Fig. 1 Funcionamiento del análisis y procesamiento de archivos de imágenes.

El orden de las imágenes previamente mencionado, es en dependencia a una propiedad de las imágenes como archivos, llamada fecha de modificación. Expresada como la variable "*TiempoModificación*" la cual es una variable tipo fecha, que determina la hora y fecha del origen del archivo. Tomando la diferencia de su valor respecto a la fecha y hora actual "*TiempoActual*" se obtiene la variable "S" tipo entero que son los segundos de diferencia entre ambas fechas, como expresado en la ecuación (5). Finalmente, dando el resultado la "x" cantidad de imágenes dentro de la estructura lineal tipo cola.

$$S = TiempoActual - FechaModificación \qquad (5)$$

El sistema de trabajo del software es un sistema cerrado de cámaras de seguridad que de manera inalámbrica o alámbrica envían las fotografías a una red para posteriormente ser guardadas en una unidad de almacenamiento dedicado al mismo sistema. Cuando se requiere se puede realizar una revisión por los usuarios definidos en la aplicación como Usuario A y Usuario B, que otorgan la ejecución de análisis de archivos y su debida identificación definida por el usuario dentro del programa DifferentFiles, que arrojará resultados dependientes de los filtros seleccionados y variables definidas.

Cabe destacar el uso de una red conectada a la web para una conexión remota desde cualquier punto basado en sólo la ejecución y monitoreo del mismo programa Fig. 2.

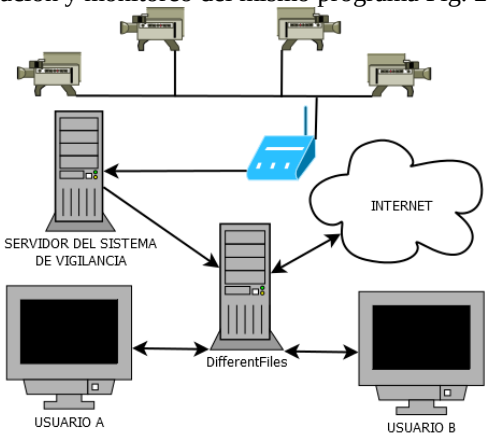

Fig. 2 Conectividad del sistema utilizado para las pruebas del software DifferentFiles.

La distinción de la identificación de imágenes consiste en analizar una "x" cantidad de imágenes con la misma información alojada en un "t" lapso de tiempo, de forma continua, consultando el peso de la imagen tomando las propiedades de las misma (png o jpg) que y analizando las variaciones de peso por su compresión y su cantidad de colores dentro del archivo, lo que indica que puede existir una diferencia dentro de un carrete de imágenes iguales o comprendidas como similares dentro de un rango de tolerancia alterará el peso de cada una en proporción a la diferencia de transición entre un cuadro a otro.

En la Fig. 3 está una comparación de dos imágenes con sus respectivos pesos en bytes, la claridad visual de un cambio es evidente también dentro el cambio del pesos debido a que la diversidad de colores dentro de la compresión de la imagen cambia radicalmente de un cuadro a otro.

Fig. 3 se aprecia la comparación de 2 imágenes tipo jpg que son diferentes.

La forma de operar del programa es ejecutándolo dentro de la carpeta que tiene contenidas las imágenes donde las analiza y pregunta filtro de peso para aplicar, teniendo 2 posibles opciones ya sea por el Filtro por porcentaje de tolerancia, o el filtro por bytes de diferencia, exponiendo su metodología y aspectos siguientes:

Filtro por porcentaje de tolerancia

El filtro de porcentaje de tolerancia se define con una estimación de tolerancia basada en un porcentaje "h" entregada por el usuario dentro del programa, posteriormente se calculan los valores de Pi_{max} (6) definiéndolo como la variable entera que ayuda a identificar las imágenes iguales dentro del rango máximo estimado de peso, a su vez Pi_{min} define el valor mínimo para declarar una imagen como igual. Los valores para definir las imágenes son aquellas que no cuenten con un peso dentro del rango estimado anteriormente, si bien su ecuación se expone como Pd_{max} y Pi_{max}.

TABLA III
DEFINICIÓN DE LAS VARIABLES PARA ECUACIÓN (6, 7, 8, 9, 10, 11, 12, 13)

Variable	Valor
Pi_{max}	Variable para definir el peso máximo que llegará a identificar una imagen como igual a la siguiente.
Pi_{min}	Variable para definir el peso mínimo que llegará a identificar una imagen como igual a la siguiente.
Pd_{max}	Variable para definir el peso máximo que llegará a identificar una imagen como distinta a la siguiente.
Pd_{min}	Variable para definir el peso mínimo que llegará a identificar una imagen como distinta a la siguiente.
h	Porcentaje de tolerancia definido por el usuario para determina una imagen a otra como iguales.
q	Valor en bytes de la tolerancia de identificación de imágenes iguales.

$$Pi_{\max} = P + \left(\frac{h}{100}\right)(p) \qquad (6)$$

$$Pi_{\min} = P - \left(\frac{h}{100}\right)(p) \qquad (7)$$

$$Pd_{\max} = P + 1 + \left(\frac{h}{100}\right)(p) \qquad (8)$$

$$Pd_{\min} = P - 1 - \left(\frac{h}{100}\right)(p) \qquad (9)$$

Filtro por bytes de diferencia

El filtro por bytes de diferencia toma el peso de cada imagen de manera individual en una variable "p" y le suma una cantidad de memoria en bytes extra "q" que el usuario le asignó haciendo esta magnitud como el valor máximo aceptable como imagen igual se obtiene de acuerdo con (10), y a su vez define un parámetro extra para restar al peso de la imagen de la variable definida por el usuario adquiriendo el valor mínimo requerido (ver (11)) como una imagen igual a la siguiente en la secuencia.

$$Pi_{\max} = P + q \qquad (10) \qquad Pi_{\min} = P - q \qquad (11)$$

Las ecuaciones hechas en el programa para discriminar las distintas solo se le añade un byte extra para poder cumplir un patrón que nunca colisione una sentencia igualatoria con otra que la haga ver diferente, teniendo así valor mínimo superior (12) y valor mínimo inferior para declarar una imagen como distinta (13).

$$Pd_{\max} = P + q + 1 \quad (12) \qquad Pd_{\min} = P - q - 1 \quad (13)$$

El diagrama de flujo del programa en Fig. 4 muestra el algoritmo que contiene el programa descrito para especificar qué tipo de filtro se usará, se continúa por su variable de tolerancia para la identificación de repetidos, una vez teniendo los parámetros dinámicos toma el último que es la fecha actual en el sistema de cómputo.

El programa después de tener la fecha actual procede con la creación de dos carpetas en el directorio donde está alojado el programa en ejecución, creados los nuevos directorios llamados DD y el RR, hace un último requisito al usuario, el cual es preguntar la extensión o tipos de archivos de las imágenes a buscar dentro del directorio. Las extensiones vienen en 3 opciones, las png, las jpg o una extra que es definirlo personalizada añadiendo los caracteres.

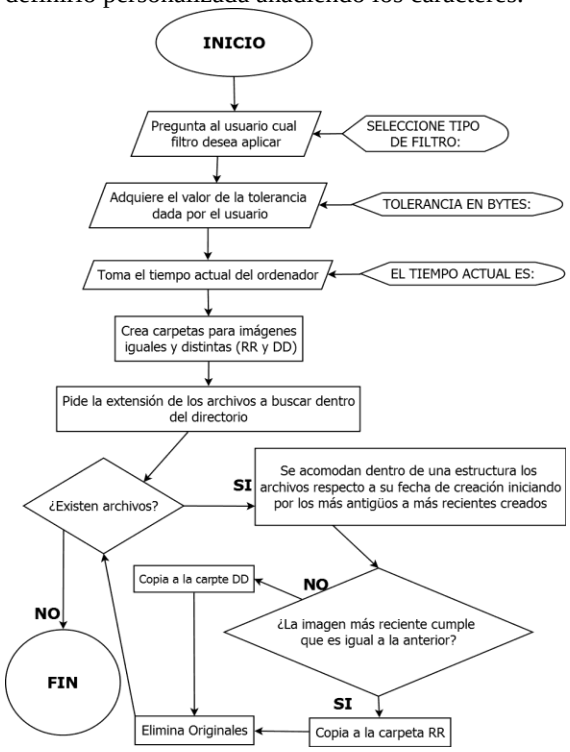

Fig. 4 Diagrama de flujo del algoritmo básico que opera el programa llamado DifferentFiles.

Las imágenes, es esencial que estén un directorio exterior de donde se encuentre el programa DifferentFiles dentro de una carpeta como se aprecia en la Fig. 5.

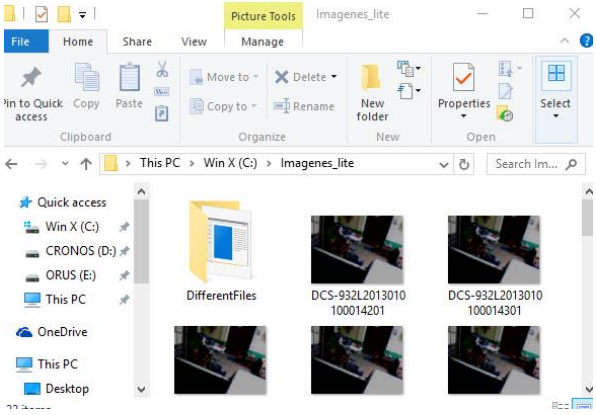

Fig. 5 forma el directorio que contiene las imágenes a usar en el software y que debe estar el ejecutable del programa dentro de un directorio (carpeta).

IV. Resultados

Utilizando la aplicación DifferentFiles, al utilizar el programa, obtuvo un ahorro de espacio superior al 60% del disco duro en el equipo de cómputo, sin embargo se piensa que dichos factores se pueden estandarizar y optimizar para un mejor rendimiento dependiendo la escena, los factores, la luz y la calidad de imagen.

Aplicando el filtro por bytes de diferencia a un segmento de muestra de un sistema de cámaras de seguridad con un total de 389 imágenes, haciendo que el programa haga el ciclo completo de búsqueda de archivos, identificación consulta de información, procesamiento de la información de cada una de las imágenes y dándole origen a un nuevo destino donde se copiarán para posteriormente eliminar las repetidas dentro de la carpeta de análisis. Los puntos previos mencionados dentro de la operación del programa tienen un lapso aproximado de tiempo de trabajo de 11.30 segundos para las 389 imágenes.

Para el análisis del funcionamiento del programa expresado en la Tabla IV y su gráfica Fig.5, se hizo una captura con 11 distintos valores y se aplicaron en las mismas 389 imágenes dentro de lo que cabe destaca que el ojo humano de forma visual, puede distinguir de una imagen a otra una diferencia a partir de una tolerancia de 35 bytes en adelante.

TABLA IV
FILTRO POR BYTES DE DIFERENCIA FIG. 5

Bytes de tolerancia	RESULTADOS AL APLICAR EN 389 IMÁGENES		
	Porcentaje liberado de memoria	Imágenes identificadas como diferentes	Imágenes identificadas como iguales
5	12%	344	45
15	34%	256	133
25	51%	192	197
35	66%	131	258
45	76%	95	294

55	83%	66	323
65	87%	50	339
75	90%	38	351
85	92%	30	359
95	92%	30	359
105	94%	24	359

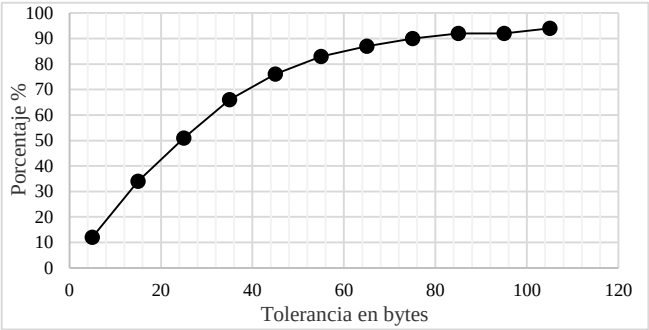

Fig. 5 define en eje "x" el filtro por bytes definidos y el eje "y" muestra el porcentaje obtenido de espacio liberado de memoria previamente ocupada.

Para la manipulación de imágenes en directorios se opta por demostrar también el filtro por porcentaje de tolerancia en las mismas 389 imágenes adquiridas por el sistema de cámaras de vigilancia. Dicho filtro funciona dela misma forma que el anterior mencionado pero mantiene una tolerancia de aproximación proporcional para poder definir la imagen como idéntica. Esto ayuda cuando se tienen unos rangos de tamaño de imagen muy diversos y no se logra cuantificar un peso acertado, también es el óptimo para manipular escenarios oscilantes, dícese que cambia su iluminación, contenido, saturación de colores, etc.

TABLA V
FILTRO POR PORCENTAJE DE TOLERANCIA FIG. 6

Porcentaje de tolerancia	RESULTADOS AL APLICAR EN 389 IMÁGENES		
	Porcentaje liberado de memoria	Imágenes identificadas como diferentes	Imágenes identificadas como iguales
0.05%	15%	329	45
0.10%	31%	256	133
0.15%	41%	192	197
0.20%	54%	131	258
0.25%	65%	95	294
0.30%	71%	66	323
0.35%	78%	50	339
0.40%	83%	38	351
0.45%	86%	30	359
0.50%	88%	30	359
0.55%	89%	24	359
0.60%	92%	33	356
0.65%	92%	30	359

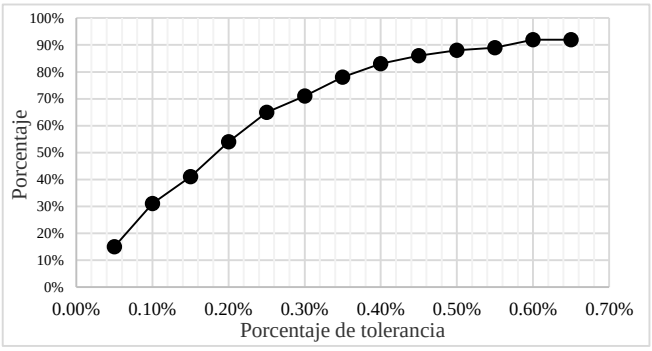

Fig. 6 Resultado de 13 ejercicios en el programa con distintas variables con el filtro por porcentaje de tolerancia en TABLA V.

Las dos clases de filtros recaban mejor información, aunque el usuario debe escoger el filtro que mejor le satisfaga en su escenario y su respectivo sistema de video seguridad. Cabe mencionar que el programa en un equipo de cómputo convencional y con 2000 imágenes tomadas de una cámara de seguridad logra consultar, procesar y entregar resultados en menos de 20 segundos, lo cual se puede considerar óptimo si se toma en cuenta que se genera 1 cuadro (imagen) por segundo en el sistema de cámaras de seguridad, además el programa será ejecutado cada vez que el usuario crea conveniente reducir el espacio ocupado de la unidad de almacenamiento.

La línea de comandos es clave en el procesamiento y direccionamiento de información para liberar los archivos definidos dentro del programa de manera rápida y sencilla descartando otras posibles librerías precargadas o más líneas de código para compilar en el código fuente.

V. CONCLUSIÓN

El desarrollo con sistemas en lenguajes básicos como CMD o C ayudan a facilitar tareas que requieren ahorro de tiempo, recursos y procesos, ayudando a resolver problemas con herramientas informáticas cada vez más presentes en los sistemas de vigilancia o manipulación de archivos.

El *fog computing* toma un rol crucial en el momento de optimizar es espacio utilizado para almacenar la información de un sistema de vigilancia y simplificación de información descartando información repetida o basura, ahorrando tiempo para la revisión de sucesos registrados con operadores humanos.

El futuro depara gran avance en el campo de la informática como también mayor accesibilidad a los medios de información, seguridad, monitoreo y bienestar pública. El programa puede avanzar aún más disponiendo de varias vertientes de utilidades y nuevos métodos para la filtración de información por archivos en sistemas de vigilancia para

Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
Digital Object Identifier: (to be inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).

mejorar la identificación de anomalías, reducir espacios utilizados y encontrar eventos hundidos en una densa neblina de información.

REFERENCIAS BIBLIOGRÁFICAS

[1] Cisco Fog Computing Solutions: Unleash the Power of the Internet of Things, Cisco Systems Inc. 2015.

[2] Tarem A., Xianglin W., Supriyo A. & Al-Sakib K. "Automated Intruder Detection from Image Sequences using Minimum Volume Sets." April 2012.

[3] Bonomi F., Milito R., Zhu J. & Addepalli S, "Fog Computing and Its Role in the Internet of Things. Cisco Systems Inc." Cisco Systems Inc. (2012) http://dl.acm.org/citation.cfm?id=2342513

[4] Henson D. H. "SPECIES VISITATION AT FREE-CHOICE QUAIL FEEDERS IN WEST TEXAS" Texas A&M University, 2006. Retrieved from: http://oaktrust.library.tamu.edu/handle/1969.1/3897

[5] Carem E., Sandra S. & James C. "Anomaly Detection for Video Surveillance Applications." McGill University, 2006.

[6] Diego G. "Sistemas de Detección de Intrusiones." GNU Free Documentation License. Julio 2003 pp. 13 – 22. Retrieved from: https://www.dgonzalez.net/papers/ids/IDS_v1.0.pdf

[7] Daniel B., Carlotta D., Zoran D., Richard M., Edgar L. & Maurizio F. "Detecting Suspicious Behavior in Surveillance Images."

[8] Saylor M. J., Slavin A., Hugues Martin J. P. "System and method for monitoring security systems by using video images." Microstrategy, Inc, 2002.