Fecha de recepción 22/10/2020
Fecha de aceptación: 29/ 04 /2021
Pp 51 – 59

Lenguaje de dominio específico para la descripción de actividades en analítica de video

Leonardo Guedez [1], Jesús Lárez [2]
leonardoguedezsuazo@gmail.com[1], jjlarez@gmail.com[2]

Escuela de Ingeniería en Informática, Universidad Catolica Andres Bello, Ciudad Guayana, Venezuela [12]

Resumen

La mayoría de los sistemas de vigilancia por video actuales aún requieren de personal que esté constantemente revisando las grabaciones de video en búsqueda de situaciones anormales u objetos de interés, haciendo que las organizaciones deban destinar personal solo a esta tarea. El presente trabajo propone el desarrollo de un lenguaje de dominio específico, llamado Hopper, que permita, con los niveles de abstracción apropiados, especificar los objetos y actividades a buscar en un sistema de analítica de video, para que estos puedan ser automáticamente detectados y se tomen las acciones pertinentes. La salida del lenguaje es un archivo de definición de objetos, actividades y acciones en un formato específico según el sistema de analítica que se usará. El desarrollo incluye la gramática del lenguaje, el traductor que genera el archivo de definición mencionado, así como un sencillo sistema de analítica que sirva de caso de estudio, el cual es capaz de detectar objetos, detectar algunas actividades, y ejecutar acciones al cumplirse una condición especificada en el programa. El trabajo se llevó a cabo bajo un desarrollo basado en el modelo en cascada. Los resultados obtenidos muestran que el lenguaje se integra bien con el sistema desarrollado, y que contiene elementos para los aspectos básicos que se esperaría tener en un sistema de analítica de video. El traductor se diseñó de manera modular, para que resulte práctico y simple agregar soporte a otros sistemas de analítica.

Palabras clave : lenguaje de dominio específico, Analítica de video, Análisis inteligente de video, Visión por computador.

Domain specific language for activity description in video analytic

Abstract

Most video surveillance systems need personnel to review the recorded videos in search of abnormal situations or objects of interest, causing organizations to dedicate staff only to this task. The present work proposes the development of a specific domain language, called Hopper, that allows, with the appropriate levels of abstraction, to specify the objects and activities to search for in a video analytics system, so that they can be automatically detected and take the appropriate actions. The language output is a file for defining objects, activities and actions in a specific format, according to the analytical system to be used. The development includes the grammar of the language, the translator that generates the definition file previously mentioned, as well as a simple analytical system that serves as a case study, which is capable of detecting objects, activities, and executing actions when a specific condition in the program is identified. Development was done using a waterfall based model. The results obtained show that the language integrates well with the developed system, and that it contains elements for the basic aspects that one would expect to have in a video analytics system. The translator was designed in a modular way, making it practical and simple to add support to other analytics systems.

Keywords: domain specific language, video analytics, intelligent video analysis, computer vision.

I. INTRODUCCIÓN

En el trabajo presentado a continuación se muestra Hopper, un lenguaje de dominio específico (*Domain Specific Language* o *DSL*, por sus siglas en inglés), de tipo declarativo, para describir las actividades y objetos que interesa detectar, así como las acciones a llevar a cabo en sistemas de analítica de video. El resultado de procesar un programa escrito en Hopper es un archivo donde se especifican las actividades, los objetos y las acciones a considerar, para que un sistema de analítica de video pueda procesarlo y cumplir con lo indicado. El formato de este archivo dependerá del sistema de analítica usado, y dicho sistema debe proveer una interfaz que pueda realizar su carga.

En el artículo se muestra brevemente el desarrollo del trabajo de grado homónimo, siendo su estructura: el planteamiento del problema, los objetivos propuestos, una breve revisión de algunos trabajos de investigación relevantes, el marco metodológico, explicación detallada de los elementos del DSL desarrollado y las etapas del proceso de traducción de sus programas, caso de estudio y programas de ejemplo con los resultados de su procesamiento, y las conclusiones del trabajo desarrollado.

II. PLANTEAMIENTO DEL PROBLEMA

El análisis de video es un aspecto crucial para muchas organizaciones, bien sea como una forma de seguridad para proteger sus espacios y zonas restringidas, o para analizar el comportamiento de las personas u objetos que se encuentran en ellas. Los sistemas clásicos de circuito cerrado de televisión y vigilancia requieren de personal que constantemente esté observando los videos, ya sean en tiempo real o grabados en un dispositivo de almacenamiento. Esto representa un aumento de costos para las organizaciones, que deben destinar personal a esta tarea. Además, las largas jornadas de trabajo pueden afectar la efectividad y desempeño cognitivo de los operadores, disminuyendo el retorno de inversión de las empresas.

En visión por computador, la detección y reconocimiento de objetos y actividades humanas han presentado un reto bastante interesante para los científicos e investigadores. Aunque se han tenido avances significativos y desarrollos notables que permiten un mejor desempeño y eficacia, aún hay muchos asuntos por resolver desde un punto de vista técnico.

Gracias a los avances en el área de visión por computador, la analítica de video, conjunto de tecnologías enfocada a automatizar tareas de seguridad tales como el monitoreo de flujos de video y ejecutar acciones asociadas a condiciones particulares y la búsqueda de contenidos específicos en repositorios de videos grabado, ha resultado ser una tecnología emergente y prometedora, con productos cada vez más útiles en dominios específicos.

En el contexto descrito, se propuso el desarrollo de un lenguaje de dominio específico que permita, con los niveles de abstracción apropiados, especificar los objetos y actividades a buscar en un sistema de analítica de video.

III. OBJETIVOS

Objetivo general:

> Desarrollar un lenguaje para la descripción de actividades a buscar en un sistema de analítica de video.

Objetivos específicos:

i. Identificar las abstracciones apropiadas para el lenguaje de descripción de actividades, a partir de los fundamentos de los sistemas de analítica de video.
ii. Definir la gramática, semántica y el procesador del lenguaje de dominio específico para analítica de video.
iii. Construir el procesador para el lenguaje de dominio específico definido.
iv. Validar el lenguaje desarrollado con un caso de estudio.

IV. TRABAJOS RELACIONADOS

a) VERSA

En **[1]**, el autor presenta un marco de trabajo de propósito general llamado VERSA, para la definición y reconocimiento de eventos en videos, bien sea en vivo o grabados. Se asume que se cuenta con un sistema de vigilancia por cámaras que envía imágenes a un sistema de analítica de video de bajo nivel para detectar, clasificar y rastrear objetos de interés. Este generará un flujo de datos en XML, específicamente en formato

CVML (CAVIAR Video Markup Language) **[2]**, un lenguaje de marcado basado en XML pero con elementos útiles para analítica de video.
Los componentes principales de VERSA, mostrados en la Figura 1, son: un analizador para el flujo de datos en CVML; un repositorio para la definición de las relaciones espaciales y temporales, y las reglas de producción usadas para verificar una relación dado un conjunto de hechos, Una "base de conocimientos" de los hechos generados por el analizador al procesar el flujo de datos; un mecanismo para monitorear y consultar la base de conocimientos para determinar si hay una coincidencia de un evento.

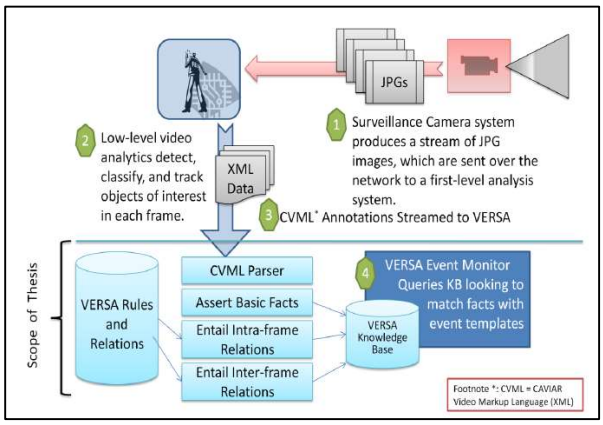

Figura 1: Esquema de alto nivel de la arquitectura de VERSA. **Fuente**: [1]

Un evento en un flujo de video es una composición de objetos identificados y las relaciones espaciales y temporales entre ellos. Por su parte, una plantilla de eventos es la especificación de un evento a buscar usando predicados específicos en el lenguaje VERSA, la tarea de reconocer un evento entonces requiere la coincidencia de una plantilla de eventos con el flujo de datos generado por el sistema de analítica, permitiendo además cierto nivel de tolerancia o incertidumbre.

b) BlazeIt

En [3] se propone un sistema para la realización y optimización de consultas realizadas sobre videos para obtener información espaciotemporal de los objetos presentes. Para realizar las consultas se utiliza el lenguaje FRAMEQL, desarrollado por los autores de dicho trabajo, el cual es un lenguaje declarativo basado en SQL, pero con elementos específicos para analítica de video exploratoria. Además, también se presenta un motor para la optimización y ejecución de las consultas realizadas.

Los autores señalan que utilizar un lenguaje similar a SQL tiene dos grandes ventajas: primero, al ser SQL un lenguaje ampliamente conocido, FRAMEQL puede ser aprendido rápidamente por sus usuarios y analistas que lo necesiten. Segundo, utilizar un enfoque declarativo permite una mayor independencia de datos, al separar la especificación del sistema de su implementación, permitiendo la extensión para nuevas optimizaciones a las consultas.

c) ADeL

Otro trabajo destacado es el presentado en [4], donde se propone el lenguaje ADeL, especialmente diseñado para especificar los escenarios y las actividades a buscar en videos, que puede integrarse en un sistema de reconocimiento de actividades que cumpla con cuatro condiciones básicas: trabajar en tiempo real, ser reactivo, producir resultados correctos y ser confiable.

Los autores explican que hay dos grandes fases en su sistema de reconocimiento de actividades propuesto:

i. La configuración del sistema, donde utilizando el lenguaje ADeL, se describen las actividades a buscar. Cada programa en ADeL se compila por separado, produciendo cada uno un archivo de librería compartida en C++, que serán cargadas por el sistema de reconocimiento de actividades al ejecutarse.

ii. El sistema de reconocimiento en tiempo de ejecución, que se encarga de extraer eventos y objetos de los datos obtenidos por los sensores, considera la incertidumbre para que en la próxima etapa de reconocimiento solo se tomen en cuenta eventos de razonable interés. Luego, estos eventos son convertidos a lo que los autores llaman un "instante lógico" creando un orden entre ellos. Finalmente, estos eventos ordenados serán la entrada de un motor de reconocimiento, que se encarga de reconocer las actividades buscadas.

En la Figura 2 se observa el esquema de la estructura del sistema de reconocimiento propuesto en [4].

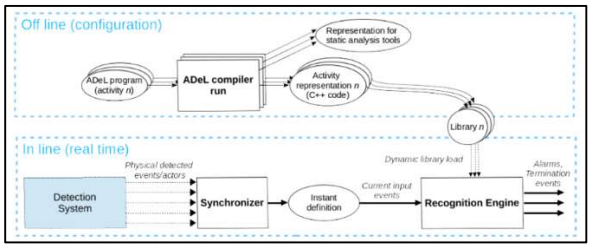

Figura 2: Estructura general del sistema de reconocimiento de actividades para ADeL.**Fuente**: [4]

En ADeL, las actividades pueden ser simples o estar compuestas por una o más sub-actividades.

V. METODOLOGÍA

Usualmente, en el diseño y construcción de lenguajes de programación se tiene un conocimiento claro de qué es lo que se quiere lograr, el fin último del lenguaje. Este nace de la idea de facilitar alguna tarea o proceso existente, proveyendo al usuario de abstracciones adecuadas para ello. Es por esto que, a diferencia de otros proyectos de software, el diseño y construcción de un lenguaje implica conocer certeramente los requerimientos del desarrollo al momento de iniciar las fases de análisis y diseño.

Debido a esto, se consideró que lo más apropiado para el trabajo era usar una metodología de desarrollo basada en el modelo de cascada, debido a que la misma se justifica en proyectos donde los requerimientos están claramente definidos y las actividades avanzan de forma lineal. Esto concuerda con la manera común de desarrollar un lenguaje: primero se detallan los fines de este, las abstracciones y facilidades que ofrece a sus usuarios con la finalidad de diseñar una especificación formal de su sintaxis, gramática y semántica para luego pasar al diseño y construcción de su procesador, sea un intérprete, compilador o un híbrido.

VI. LENGUAJE DESARROLLADO: HOPPER

La solución planteada en este trabajo consiste en un lenguaje de dominio específico para la búsqueda de actividades y objetos en analítica de video, llamado Hopper. El lenguaje permite especificar qué actividades y objetos de interés se deben buscar en los videos, así como especificar las propiedades que deben cumplir los objetos para ser detectados. También, permite indicar qué acciones ejecutar al cumplirse una condición.

Naturalmente, se requiere de un traductor para que los programas escritos en el lenguaje puedan ser usados. El programa es traducido a una representación intermedia en un formato independiente del sistema de analítica, representación que luego es consumida y procesada por este. Adicionalmente, el traductor puede corroborar que las actividades, propiedades, acciones y objetos especificados en el programa sean válidos según las capacidades del sistema de analítica, mediante un archivo que este último debe proveer en un formato específico, llamado archivo de manifiesto.

En este apartado se describe el lenguaje desarrollado, el proceso de traducción de sus programas, la arquitectura del traductor, y finalmente se muestran algunos programas de ejemplos escritos en el lenguaje, junto con el resultado de ejecutarlos en un sistema de analítica de video.

El lenguaje Hopper cuenta con los siguientes elementos principales para sus programas:

- Detección de actividades, mediante la declaración *detect activity* seguida de, entre comillas dobles, el nombre de la actividad a detectar. Opcionalmente también se puede indicar la zona y las cámaras en donde buscar la actividad.
- Detección de objetos, mediante la declaración *track object* seguida de, entre comillas dobles, el nombre de la clase de objeto a detectar. Opcionalmente se pueden indicar la cantidad mínima o máxima de objetos a detectar, las zonas y cámaras en donde buscar, y las propiedades que deben cumplir los objetos de interés.
- Ejecutar acciones condicionales asociadas a objetos. Por ejemplo, tomar una captura de pantalla al detectarse cinco o más personas.
- Ejecutar acciones condicionales asociadas a actividades. Por ejemplo, generar una alerta al detectarse la actividad “corriendo”.

- Declaración de variables, para utilizarlas en otras partes del programa. Por ejemplo, se puede guardar el perfil de un sospechoso (persona con camisa verde y pantalón azul) en una variable, para utilizarla luego en una sentencia *track object*.
- Especificar las cámaras y zonas en donde buscar las actividades u objetos.

A su vez, el lenguaje permite comentarios para documentar el código, utilizando el carácter punto y coma (;), que indica un comentario de una línea.

La estructura de los elementos descritos se muestra a continuación:

- *detect activity* "actividad" [*inzone* "zona"] [*from camera* (*name* | *ip* | *number*)*:* "valor"]
- *track object* [min] [max] [*with* "nombre-propiedad" "valor-propiedad"] [*inzone* "zona"] [*from camera* (*name* | *ip* | *number*)*:* "valor"] [incr contador]
- *when* variable > numero *do* accion()
- *on activity* "actividad" *do* accion()
- nombre-variable = valor

Las cursivas denotan palabras reservadas por el lenguaje, los corchetes indican elementos opcionales, y las palabras separadas por barras verticales indican que solo una de dichas palabras reservadas puede usarse en un momento dado. Se permite el carácter '-' para los identificadores, tanto de variables como de acciones.

a) Proceso de traducción

Una vez escrito el programa en el lenguaje Hopper, este debe traducirse para generar lo que se conoce como un Archivo de Definición de Objetos, Actividades y Acciones (DOAA). Este archivo estará en un formato específico, según lo que soporte el sistema de analítica de video destino. El sistema de analítica procesará este archivo para registrar qué actividades y objetos buscar, además de las acciones a llevar a cabo.

En general, el proceso de traducción sigue los siguientes pasos:

i. Se invoca al traductor, indicándole el archivo de entrada, el nombre del sistema de analítica a utilizar, y el formato del archivo a generar.
ii. Se lleva a cabo la etapa de análisis o front-end del traductor, donde se realiza el análisis léxico, sintáctico y semántico, reportando al usuario cualquier error sintáctico que pudiera haber, abortando el resto del proceso en caso de que esto ocurra.
iii. Se genera la representación intermedia.
iv. La etapa de síntesis o back-end utiliza esta representación intermedia para generar el archivo DOAA, según el formato indicado en el paso i.

En la Figura 3 se observa un esquema del proceso de traducción.

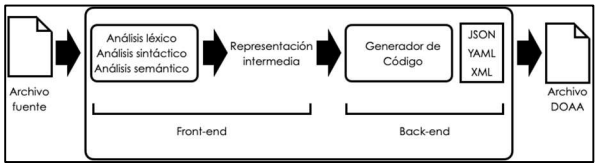

Figura 3: Esquema general del proceso de traducción de programas en Hopper. **Fuente:** Elaboración propia.

Es importante notar el elemento llamado Generador. Ese es el que encapsula, según el formato indicado, la generación en sí del archivo DOAA. Este generador variará dependiendo del formato esperado por el sistema de analítica con que se trabajará; en la imagen se muestran tres formatos, JSON, YAML, XML, ya que esos eran los aceptados por el sistema del caso de estudio desarrollado, como se explica en el siguiente apartado.

b) Caso de estudio

Se desarrolló un sistema de analítica de video simple para comprobar la efectividad que puede tener el lenguaje en dar a los usuarios abstracciones relevantes del dominio. El sistema soporta la detección de dos actividades (caminar y correr), detección de objetos como persona y automóvil, así como ejecutar ciertas acciones al cumplirse una condición. Los formatos para el archivo DOAA soportados por este sistema son JSON, YAML y XML.

En la Figura 4 se muestra el proceso completo de uso del lenguaje en el sistema, desde la traducción, hasta la detección y la generación de eventos.

Para iniciar el sistema, se le invoca pasándole como argumento la ruta del archivo DOAA. Se ejecutará un módulo encargado específicamente

de procesar el archivo y registrar qué es lo que se debe buscar. Luego esto pasará al módulo de detectores, donde se detectarán las actividades y objetos en el video, y se le asignará un identificador único a cada instancia de un objeto (ej.: si hay tres personas detectadas, cada una tendrá un identificador único); finalmente se mostrará el resultado del procesamiento por pantalla.

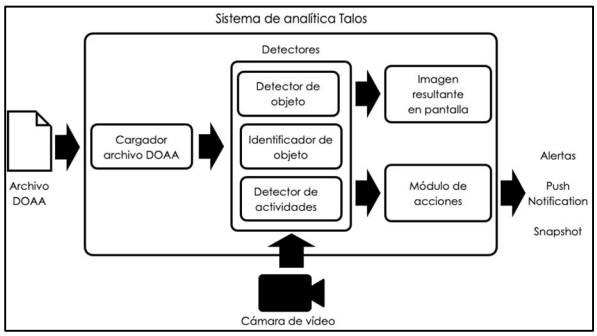

Figura 4: Esquema del proceso de funcionamiento del sistema de analítica de video. **Fuente**: Elaboración propia.

En caso de cumplirse una condición asociada a un objeto o a una actividad, se llamará al módulo de acciones, que tiene el código que finalmente la ejecutará. Para el caso de estudio se consideraron tres acciones:

- *Alert*, para alertar que ya se cumplió una condición, mostrando un mensaje por pantalla.
- *Push notification*, que enviará una notificación push con la imagen capturada en el momento en que se cumplió la condición, y una descripción, a una aplicación móvil para un teléfono inteligente.
- *Snapshot*, que tomará una captura de pantalla del momento en que se cumplió la condición, y la guardará en el sistema de archivos.

c) Validación del programa fuente

En adición a esto, al momento de traducir el programa fuente se tiene la opción de validar que lo especificado en este cumpla con lo soportado por el sistema de analítica de video; es decir, validar que el usuario no indique cosas (actividades, objetos, acciones, etc.) que el sistema de analítica a utilizar no soporte.

Esta validación se hace gracias a un archivo de manifiesto, en donde se deben especificar qué elementos son soportados por el sistema de analítica, tales como las actividades, los objetos y sus propiedades, las acciones y la cantidad de argumentos de cada una, etc.

Para validar el programa fuente, se pasa la ruta del archivo de manifiesto como argumento de la opción -c (o --check en su forma larga). El archivo de manifiesto debe estar en formato JSON.

VII. PRUEBAS Y RESULTADOS

En este apartado se muestran algunos programas de prueba escritos en Hopper, así como el resultado de ejecutarlos en el sistema de analítica desarrollado.

a) *Rastreo de objetos*

El siguiente programa rastrea personas (detecciones de la clase *person*) sin ningún tipo de filtro, es decir, sin indicar ninguna propiedad en específico deseada:

track “person”

El resultado de traducir este programa a un archivo DOAA y cargar este en el sistema de analítica desarrollado es el mostrado en la Figura 5.

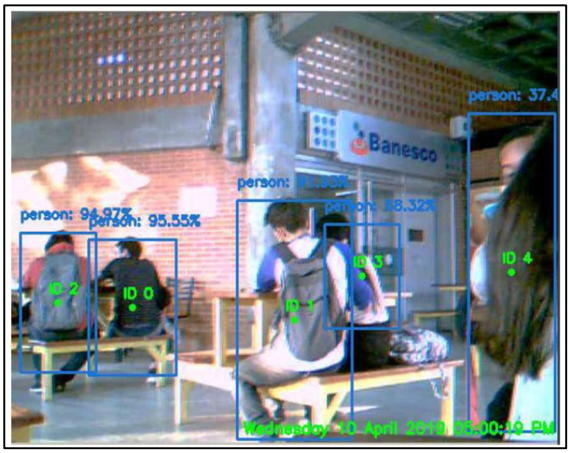

Figura 5: Resultado del programa para buscar personas en el sistema de analítica desarrollado. **Fuente**: Elaboración propia.

Se observa que, efectivamente, se detectaron todas las personas presentes en la escena, encerradas en un cuadro delimitador que incluye, en el centro, el identificador único dado a cada

instancia, y arriba a la izquierda el porcentaje de certeza de pertenecer a la clase indicada. En la esquina inferior derecha también se tiene la fecha y hora exacta de ejecución del sistema de analítica.

b) *Rastreo de objetos especificando propiedades*

El siguiente programa es similar al anterior, pero ahora se indica buscar solo personas con camisa roja (propiedad *shirt*, valor *red*):

```
track “person” with “red” “shirt”
```

En la Figura 6 se observa el resultado de procesar el archivo DOAA generado por la traducción.

Figura 6: Resultado de buscar personas con camisa roja en el sistema de analítica. **Fuente:** Elaboración propia.

Nótese que, aunque claramente hay dos personas con camisa roja en la escena, la persona de la izquierda tiene la mayor parte visible del torso cubierto por el cabello, por lo que el sistema no la considera en la detección.

c) *Rastreo de objetos usando variables, contadores y condiciones*

En el siguiente programa se hace uso de otros elementos del lenguaje útiles para la analítica de video, como contadores y ejecución de acciones, además de variables.

```
sospechoso = target “person” with “red” “shirt”

umbral = 3

track sospechoso incr contador

when contador > umbral do alert(“La cantidad de personas es mayor al umbral!”)
```

Nuevamente, el programa rastrea personas con camisa roja, pero ahora utiliza una variable para guardar el perfil de la persona buscada, y otra para guardar un umbral máximo de personas permitidas en la escena. Al detectar una persona con camisa roja, incrementará un contador. Cuando este contador supere el valor de la variable umbral, se ejecutará la acción de alerta. El resultado de la ejecución, similar al programa del apartado anterior, se muestra en la Figura 7.

Nótese que la sintaxis del lenguaje sigue un patrón declarativo, donde se le da más importancia a lo que se quiere lograr, y no tanto al cómo.

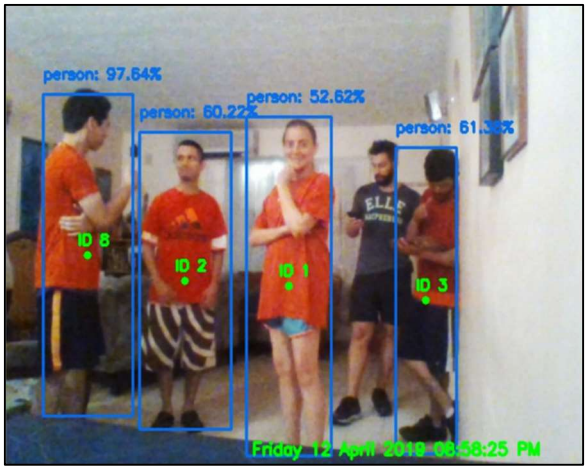

Figura 7: Resultado de buscar personas con camisa roja en el sistema de analítica, con un programa que utiliza variables, contadores y condiciones. **Fuente:** Elaboración propia.

Obsérvese cómo la persona con camisa gris es ignorada, mientras que las personas de camisa roja son marcadas. En la Figura 8, al final, se observa la alerta, que muestra un mensaje por pantalla:

```
Leonardo@DESKTOP-ECIE775 MINGW64 ~/cvsys (master)
$ python main.py -f config.json
[INFO] loading model...
[INFO] starting video stream...
ALERT: La cantidad de personas es mayor al umbral!
```

Figura 8: Alerta generada al haber una cantidad de personas mayor al umbral. **Fuente:** Elaboración propia.

d) *Detección de actividades*

Ahora se muestra un programa de ejemplo para detectar actividades, en este caso la actividad "caminar":

detect activity "walking"

Cada vez que se detecte la actividad caminar, se mostrará un mensaje en pantalla indicándolo en la parte superior izquierda de la ventana, como se observa en la Figura 9.

Figura 9: Resultado de buscar la actividad de caminar en el sistema de analítica. **Fuente:** Elaboración propia.

e) *Detección de actividades con acciones condicionales*

Finalmente, se muestra la ejecución de acciones al detectarse una actividad. Al igual que el programa anterior, se busca la actividad "caminar", pero ahora al detectarse, se generará una notificación push que llegará a un teléfono inteligente con una aplicación móvil nativa desarrollada para el caso de estudio.

detect activity "walking"

on activity "walking" do push-notification("¡Hay alguien caminando en la escena!")

El resultado se muestra en la Figura 10.

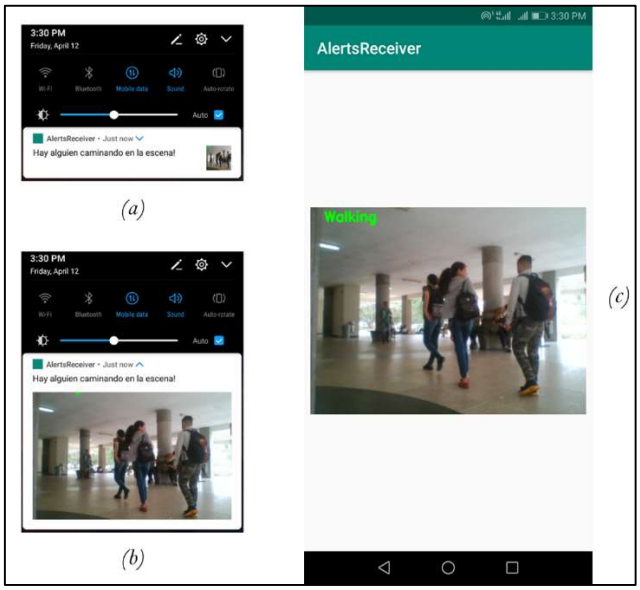

Figura 10: Uso de mecanismo de notificaciones push para acciones condicionales en la búsqueda de actividades. (a) Notificación push recibida en el dispositivo móvil. (b) Notificación push expandida, mostrando el mensaje y la imagen. (c) Interfaz principal de la aplicación con la imagen recibida por la notificación. **Fuente:** Elaboración propia.

VIII. CONCLUSIONES

El avance en técnicas para visión por computador y aprendizaje de máquina ha aumentado y mejorado las capacidades de los sistemas de analítica de video, especialmente en el área de vigilancia y seguridad. Sin embargo, la mayoría de estos aún requieren de personal frente a las pantallas, monitoreando el video en busca de sujetos y actividades sospechosas.

La solución expuesta en este trabajo es un lenguaje de dominio específico con abstracciones para especificar las actividades y objetos a buscar en el video, además de las acciones a ejecutar en cada caso.

La solución planteada en este trabajo fue influenciada por el producto de la revisión de investigaciones previas,

Se expusieron los elementos sintácticos del lenguaje y el proceso de traducción de sus programas, también se mostraron ejemplos de programas escritos en el lenguaje, junto a los resultados de ejecutarlos en un sistema de analítica desarrollado, y los resultados mostraron

que, efectivamente, el lenguaje cuenta con elementos que resultan útiles a los usuarios a quienes va destinado.

La arquitectura modular del procesador del lenguaje permite extenderlo fácilmente para agregar soporte a nuevos elementos deseados. Posibilitando por ejemplo agregar soporte para generación de estadísticas que puedan ser útiles.

El código fuente para el traductor del lenguaje Hopper puede encontrarse en el siguiente repositorio:

https://github.com/leolas95/hopper.git

El código fuente para el sistema de analítica de video desarrollado se encuentra alojado en el siguiente repositorio:

https://github.com/leolas95/talos.git

IX. REFERENCIAS

[1] S. O'Hara, «VERSA – Video Event Recognition for Surveillance,» Omaha, 2008.

[2] T. List y R. B. Fisher, «CVML - An XML-based computer vision markup language,» Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04), vol. 1, nº 01, pp. 789-792, 2004

[3] D. Kang, P. Bailis y M. Zaharia, «BlazeIt: Fast Exploratory Video Queries using Neural Networks,» Stanford InfoLab, 2018.

[4] I. Sarray, A. Ressouche, S. Moisan, J.-P. Rigault y D. Gaffé, «An Activity Description Language for Activity Recognition,» de IEEE International Conference on Internet of Things, Embedded Systems and Communications, Gafsa, 2017.