Universidad Autónoma de Querétaro Facultad de Informática

Lenguaje de descripción de eventos para análisis de video a través de un sistema distribuido multicámara.

Tesis

Que como parte de los requisitos para obtener el Grado de Doctora en Ciencias de la Computación

Presenta:
M.C.C. Selene Ramírez Rosales

Dirigido por:
Dr. Hugo Jiménez Hernández

Sinodales

Dr. Hugo Jiménez Hernández
Presidente

Dra. Sandra Luz Canchola Magdaleno
Secretario

Dr. Fausto Abraham Jacques García
Vocal

Dra. Gabriela Xicoténcatl Ramírez
Suplente

Dra. Ana Marcela Herrera Navarro
Suplente

Centro Universitario, Querétaro, Qro.
Mayo 2025

RESUMEN

El aumento en la demanda de sistemas de vigilancia ha impulsado la búsqueda de soluciones capaces de prevenir y predecir eventos cada vez más complejos. La integración de estos sistemas presenta diversos retos, como el acceso a entornos, variedad de sensores y cámaras disponibles en la actualidad, además la necesidad de un enfoque solido para la detección de movimiento e inferencia de actividades. En este contexto surge SEL(Semantic Event Language) un lenguaje de alto nivel para el análisis de video en entornos distribuidos de múltiples cámaras, proporcionando una visión más completa de las actividades en múltiples ubicaciones, diseñada para entornos distribuidos de vigilancia con multicámara. SEL ofrece un marco estructurado, expresivo y accesible para modelar y detectar actividades a partir de primitivas de movimiento, considerando relaciones espaciales y temporales. Este lenguaje de alto nivel permite representar eventos de forma coherente al integrar datos provenientes de múltiples fuentes y ángulos, mejorando la escalabilidad, la tolerancia a fallos y la toma de decisiones en entornos complejos. Su aplicación en sistemas multicámara facilita la definición y análisis de actividades complejas, optimizando la comunicación entre componentes y proporcionando una visión unificada del entorno. SEL se presenta como una solución flexible y efectiva para el análisis de video en aplicaciones como la seguridad pública, la gestión de infraestructura y las ciudades inteligentes.

Palabras claves: Lenguaje inferencia de actividades, lenguajes distribuidos, primitivas de movimiento, sistema de vigilancia.

SUMMARY

The increasing demand for surveillance systems has driven the search for solutions capable of preventing and predicting increasingly complex events. Integrating these systems presents several challenges, such as limited access to specific environments, the diversity of available sensors and cameras, and the need for a robust approach to motion detection and activity inference. In this context, SEL(Semantic Event Language) emerges as a high-level language for video analysis in distributed multi-camera surveillance environments, providing a more comprehensive view of activities across multiple locations. SEL offers a structured, expressive, and accessible framework for modeling and detecting activities using motion primitives, considering spatial and temporal relationships. This high-level language enables coherent event representation by integrating data from various sources and viewpoints, enhancing scalability, fault tolerance, and decision-making in complex environments. Its implementation in multi-camera systems facilitates the definition and analysis of complex activities, optimizes communication between components, and delivers a unified view of the environment. SEL stands out as a flexible and effective solution for intelligent video analysis in applications such as public safety, infrastructure management, and smart cities.

Keywords: Activity inference language, distributed languages, movement primitives, surveillance system.

AGRADECIMIENTOS

Quiero expresar mi agradecimiento a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) y a la Universidad Autónoma de Querétaro (UAQ), en especial a la Facultad de Informática, así como a todas las personas que estuvieron involucradas en el transcurso de mis estudios de doctorado. Agradezco sinceramente a cada una de las personas que compartieron sus conocimientos, lo que hizo posible la finalización de esta etapa académica. Quiero destacar especialmente la guía y el apoyo brindados por mi asesor a lo largo de este proyecto el Dr. Hugo Jiménez Hernández.

TABLA DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Capítulo 1

Introducción

Actualmente, los sistemas de vigilancia están orientados principalmente al reconocimiento de acciones en videos han mostrado un gran crecimiento en el abordaje de temas como: la detección de eventos anómalos, el reconocimiento de eventos y acciones, y la comprensión de actividades (Yin, Sinnott, y Jayaputera, 2024).

El reconocimiento y monitoreo de actividades y comportamientos tiene diferentes objetivos, enfocados a las actividades realizadas en los entornos, para el monitoreo de personas mayores, pacientes o bebés. Así, como el monitoreo de tráfico de automóviles y peatones, seguimiento de actividad humana, conteo de personas, etc. Estos sistemas están diseñados para proporcionar una comprensión y análisis de actividades basadas en el conocimiento adquirido y operar sobre lo observado en diferentes contextos, ya sea en carreteras, aeropuertos, escuelas, entre otros (Tran, Vu, Vo, Nguyen, y Nguyen, 2022).

Comprender cuales y como ocurren los eventos en los diferentes entornos es crucial para la toma de decisiones, especialmente en una área llena de diferentes enfoques y métodos diferentes. El análisis de imágenes juega un papel fundamental en este proceso, ya que ofrece una perspectiva única y detallada, permitiendo observar, interpretar y describir con mayor precisión los sucesos que ocurren a nuestro alrededor. En este sentido, el análisis de imágenes se presenta como una herramienta invaluable para el monitoreo y la detección de actividades en diferentes entornos (Amosa y cols., 2023; Shidik y cols., 2019).

La inferencia de actividades mediante sistemas de vigilancia permite definir estructuras basadas en movimientos y comportamientos de los objetos analizados.

Al observar y evaluar comportamientos a través de técnicas de visión por computadora, para desarrollar modelos de precisión las acciones humanas (Shidik y cols., 2019). La capacidad de inferir y entender las actividades humanas a partir del movimiento capturado en sistemas de visión genera la posibilidad de mejorar la efectividad de los sistemas de vigilancia y sus aplicaciones (Sun y cols., 2023).

Nuestro entorno está interconectado mediante una compleja red de componentes, que incluye sensores, cámaras, entre otros, los cuales trabajan de forma conjunta para proporcionar información y potenciar las capacidades de los sistemas (Yang, Cai, Liu, Ke, y Wang, 2023; Amosa y cols., 2023).

La interconexión de estos componentes es crucial, ya que permite la recopilación y análisis de datos de manera accesible, lo cual resulta esencial para la automatización efectiva de los procesos (Rakhimov, Mamadjanov, y Mukhiddinov, 2020). La integración de estos sistemas mejora la seguridad y organización, así como la innovación y la adaptación tecnológica en diversos campos (Morshed, Sultana, Alam, y Lee, 2023).

Los sistemas tradicionales de han utilizado cámaras individuales, lo cual resulta de gran utilidad en espacios reducidos o pequeños donde un solo punto de vista es suficiente para cubrir el área de interés, sin embargo, en contextos más complejos o de mayor escala esta solución resulta limitada, ya que no permite una cobertura completa ni un seguimiento eficiente de múltiples puntos de interés, por lo cual un sistema distribuido integra múltiples fuentes de información (Olagoke, Ibrahim, y Teoh, 2020)

La transición de los sistemas de vigilancia de modelos centralizados a sistemas distribuidos representa un cambio significativo en la gestión y el análisis de la información. Inicialmente, los sistemas de vigilancia dependían de una única unidad de control central que procesaba la información proveniente de cámaras y sensores. Sin embargo, el aumento en la complejidad y el volumen de datos generados puso de manifiesto las limitaciones de estos sistemas en términos de escalabilidad y eficiencia (Elharrouss, Almaadeed, y Al-Maadeed, 2021).

Los sistemas distribuidos ofrecen una solución a estas limitaciones, mejo-

rando la gestión de recursos y proporcionando una mayor flexibilidad para adaptarse a las necesidades actuales (Abad, Ortiz-Holguin, y Boza, 2021; Elharrouss y cols., 2021). Además, estos sistemas facilitan la comunicación entre sus componentes, facilitando el envío, análisis y recepción de información (Berger y Lu, 2022). Además, que permite un monitoreo desde diferentes ángulos, ya que al contar con múltiples perspectivas se reduce el riesgo de puntos ciegos, se incrementa la precisión en la interpretación de los movimientos y se facilita el seguimiento de personas u objetos en movimiento a través de distintos espacios.

La principal motivación para el uso de un sistema distribuido en un sistema de vigilancia radica en la posibilidad de compartir recursos, evitando la centralización y permitiendo un acceso más amplio a la información. Esto se logra mediante la integración de diversas herramientas para el análisis, detección e inferencia de actividades, que incluyen nuevos tipos de sensores, cámaras y dispositivos especializados (Zhang y cols., 2022; Ramirez-Rosales y cols., 2024).

Por otro lado, los sistemas de vigilancia distribuida enfrentan importantes desafíos: el incremento en el tiempo para él envió de la información y como la ausencia de estándares para implementar diversos algoritmos, así como la sincronización de múltiples dispositivos, y la seguridad en la transmisión de datos(Yin y cols., 2024).

A pesar de estos retos, los sistemas distribuidos ofrecen numerosas ventajas, como la tolerancia a fallos asegura la continuidad operativa incluso frente a errores individuales.

Estas características otorgan a los sistemas una mayor robustez y eficiencia, permitiéndoles manejar grandes volúmenes de datos y usuarios concurrentes, al tiempo que optimizan la gestión de la información y la infraestructura de red (Yin y cols., 2024).

Los sistemas distribuidos son fundamentales en la infraestructura tecnológica moderna, ya que ofrecen robustez y eficiencia en el manejo de datos y procesos. Paralelamente, los lenguajes formales desempeñan un papel crucial en la definición precisa de estos sistemas, al proporcionar un marco estructurado pa-

ra describir con exactitud las operaciones, comportamientos y comunicaciones dentro de ellos.

La combinación de una arquitectura distribuida y un lenguaje formal permite construir sistemas de video vigilancia con mayor complejidad, donde un entorno a mayor escala es capaz de analizar e inferir actividades o eventos de manera sencilla, mediante la coordinación entre dispositivos y procesamiento datos en paralelo.

En este contexto, la integración de sistemas distribuidos y lenguajes formales no solo facilita la definición y verificación de sistemas robustos, sino que también abre nuevas posibilidades en áreas emergentes, como el análisis y reconocimiento de actividades. Los lenguajes formales, al proporcionar una base estructurada para modelar operaciones complejas, pueden extenderse hacia la representación semántica de datos, lo que resulta relevante en escenarios dinámicos y diversos.

Las investigaciones recientes han explorado el uso de lenguajes formales y procesamiento de textos para el análisis y reconocimiento de actividades(Alevizos, Skarlatidis, Artikis, y Paliouras, 2017). Las características de los métodos semánticos han demostrado ser de gran utilidad para abordar problemas relacionados con la variabilidad de escenarios, elementos y métodos.

El uso de la semántica permite modelar actividades similares en diferentes contextos, entornos, campos de visión, formas y objetos. Los lenguajes formales ofrecen ventajas significativas, como una baja complejidad y una mayor expresividad para el modelado de actividades. Esto los convierte en herramientas efectivas para abordar la complejidad inherente al modelado e inferencia de actividades (Morshed y cols., 2023).

Los lenguajes formales, gracias a su gramática y reglas estructuradas, proporcionan un marco sólido para representar estructuras de información dentro de los sistemas. Esto simplifica el proceso de análisis y permite asignar un significado preciso a las actividades en diversos contextos y escenarios (Alur, Stanford, y Watson, 2023).

Además, los lenguajes formales son una gran ayuda para el aprendizaje

técnico, ya que permiten expresar estructuras complejas a través de gramáticas. Esto, a su vez, mejora nuestra comprensión e interpretación de la información (Alur y cols., 2023).

La utilización de un lenguaje formal o de alto nivel ofrece una herramienta eficaz para la comunicación e interpretación de información, permitiendo representarla a través de un sistema distribuido multicámara y facilitando los procesos de modelado e inferencia de comportamientos en diferentes contextos. En un lenguaje formal, el usuario emplea reglas gramaticales y sintácticas definidas para crear instrucciones que permitan modelar e inferir comportamientos mediante el uso de primitivas de movimiento.

Basado en lo anterior, se propone un lenguaje de alto nivel distribuido multicámara, el cual es esencial en la actualidad para el diseño y operación de sistemas de vigilancia distribuidos. Este tipo de lenguaje permite una comunicación y coordinación mucho más efectiva entre los diferentes componentes que están geográficamente dispersos en varias ubicaciones.

Estos lenguajes están diseñados para el manejo de las complejidades de los entornos distribuidos, proporcionando estructuras y herramientas que facilitan la creación de aplicaciones robustas y escalables.

1.1 Planteamiento del problema

En la actualidad, los sistemas centralizados de vigilancia y monitoreo se han visto limitados ya que no han logrado manejar de manera efectiva la complejidad y diversidad de los diferentes escenarios.

Estos sistemas enfrentan restricciones significativas en términos de escalabilidad, capacidad de respuesta y flexibilidad para adaptarse a diversos entornos. La centralización impone obstáculos en el procesamiento de información, una capacidad limitada para integrar múltiples escenarios. Esto ha incrementado la demanda por soluciones que no solo sean robustas, sino también fáciles de usar y eficientes.

En este contexto, la transición hacia sistemas distribuidos y avanzados se

convierte en una necesidad para superar estas limitaciones.

La creciente necesidad de monitorear y analizar actividades en varios escenarios simultáneamente ha evidenciado las limitaciones de los sistemas centralizados. Con los avances tecnológicos, la demanda de soluciones de vigilancia ha aumentado la búsqueda de sistemas capaces de interpretar comportamientos y actividades en entornos distribuidos con precisión y eficiencia. Sin embargo, los sistemas actuales no logran satisfacer estas exigencias debido a la dificultad de integrar y procesar datos de múltiples cámaras.

La necesidad de soluciones óptimas para su implementación en sistemas distribuidos se centra en permitir un sistema remoto capaz de modelar e inferir actividades en diversos escenarios. Por lo cual, se propone un lenguaje de alto nivel basado en primitivas de movimiento para escenarios distribuidos, para el modelado e inferencia del desplazamiento e interpretación a través de múltiples cámaras.

El lenguaje facilita el proceso de molado e inferencia de actividades de forma eficiente y precisa en entornos distribuidos. Ayudando a la comunicación y la definición de actividades entre múltiples escenarios. Este enfoque promueve una comunicación fluida y una definición clara de las actividades, siendo accesible para los usuarios finales, incluidos aquellos con conocimientos limitados en visión por computadora, lo cual amplía su aplicabilidad en diversos programas y contextos.

1.2 Justificación

Investigaciones previas indican que utilizar gramáticas o un lenguaje de alto nivel basado en primitivas de movimiento para modelar comportamientos es tanto útil como fácil de utilizar para un usuario final. Esto se presenta como una alternativa efectiva para obtener información y estructuras que faciliten el modelado e inferencia de comportamientos

El uso de un lenguaje de alto nivel en un sistema distribuido facilita la obtención de información en diversos escenarios, permitiendo además el intercambio

de datos entre los componentes interrelacionados.

Por lo tanto, un lenguaje de alto nivel para configurar sistemas de visión amplía el alcance para los usuarios y mejora la exactitud en la descripción de los comportamientos. El uso de un lenguaje en un sistema distribuido multicámara permite una mayor cobertura de los escenarios, lo que facilita la identificación de actividades complejas o interacciones, de acuerdo con la dinámica de desplazamiento de los objetos en los diferentes entornos.

Contar con un lenguaje de alto nivel para la configuración de sistemas de visión multicámara amplía las posibilidades para los usuarios, al facilitar la descripción de comportamientos y permitir un mayor alcance en los escenarios observados. Esto contribuye al análisis de actividades complejas e interacciones, considerando la dinámica de desplazamiento de los objetos en distintos entorno.

1.3 Hipótesis

Si se desarrolla un lenguaje de descripción de eventos basado en primitivas de movimiento, será posible modelar e identificar actividades en escenarios a través de un sistema distribuido multicámara.

1.4 Objetivo general

Desarrollar un lenguaje de descripción de eventos para análisis de video en sistemas distribuidos multicámara, basado en primitivas de movimiento, utilizando una gramática libre de contexto para modelar e identificar actividades en escenarios diversos.

1.5 Objetivos particulares

- Definir las primitivas de movimiento necesarias para describir los eventos en un sistema multicámara.
- Establecer la gramática del lenguaje de descripción de eventos propuesto.

- Diseñar la arquitectura de un sistema distribuido capaz de procesar datos de múltiples cámaras.
- Implementar el sistema de comunicación entre los componentes del sistema distribuido, asegurando una correcta integración de las cámaras y el procesamiento de datos.
- Desarrollar un conjunto de programas, utilizando el lenguaje definido, para identificar y modelar actividades en diversos escenarios con múltiples cámaras.
- Validar experimentalmente el sistema en escenarios reales, para evaluar su capacidad de modelar e identificar actividades con precisión.

Capítulo 2

Estado del arte

En este capítulo, se presenta una revisión de los antecedentes y fundamentos teóricos necesarios para comprender el desarrollo y aplicación de un lenguaje para el modelado e inferencia de actividades en un sistema distribuido multicámara.

La detección de actividades ha sido un área de estudio en crecimiento dentro del campo de la visión por computadora. Tradicionalmente, los métodos para la detección de actividades se han basado en técnicas de procesamiento de imágenes, como la detección de bordes, segmentación y el seguimiento de objetos (Dutta, Boongoen, y Zwiggelaar, 2025). Estas técnicas iniciales permiten identificar y rastrear objetos en movimiento dentro de una secuencia de vídeo, proporcionando una base para la detección de actividades más complejas (Gao y cols., 2023).

La investigación sobre el reconocimiento de acciones en vídeo es un área establecida y en crecimiento debido a la amplia gama de entornos, situaciones y variables que pueden existir. En los últimos años, los sistemas de vigilancia automática se han especializado en el reconocimiento de actividades por vídeo han progresado significativamente (Devanne y cols., 2015; Kumar y Kumar, 2023).

Estos avances incluyen mejoras en la detección y seguimiento de objetos, así como en el análisis del comportamiento humano. También la integración de múltiples fuentes de datos, como cámaras y sensores y el desarrollo de modelos con mayor precisión basados en aprendizaje profundo que permiten una interpretación más robusta de actividades, incluso con múltiples objetos (Dutta y cols., 2025).

Con el avance de los algoritmos de aprendizaje automático, se introdujeron métodos más sofisticados para modelar e inferir actividades. Los modelos ocultos de Markov (HMM) y las máquinas de soporte vectorial (SVM) fueron algunas de las primeras técnicas empleadas para clasificar actividades humanas basadas en características extraídas de secuencias de vídeo (Kulsoom y cols., 2022).

Estos enfoques principalmente trabajan a partir de secuencias temporales de datos y patrones de movimiento. Sin embargo, estos enfoques presentan limitantes significativas al trabajar con grandes volúmenes de datos y ampliar la escalabilidad (Dutta y cols., 2025).

Uno de los grandes desafíos en el análisis de video es interpretar correctamente las actividades en su contexto. Esto no solo significa identificar movimientos individuales, sino también entender cómo interactúan los objetos y la dinámica general de la escena (Simonyan y Zisserman, 2014). Esta complejidad añade una capa extra de dificultad al modelar e inferir actividades, ya que se necesitan enfoques que puedan integrar de manera coherente y significativa la información espacial y temporal.

Para hacer que la detección de actividades sea más efectiva, la adopción de técnicas de aprendizaje profundo ha revolucionado la detección de actividades. Las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN) han demostrado ser extremadamente efectivas para la extracción de características y la interpretación de secuencias de vídeo de manera automatizada (Wang, Liu, Zhang, Chen, y Guan, 2017).

Las Redes Neuronales Convolucionales (CNN) se utilizan para extraer características espaciales de los cuadros de vídeo, mientras que las Redes Neuronales Recurrentes (RNN), en particular las variantes de memoria a largo plazo (LSTM), se emplean para capturar dependencias temporales a lo largo de las secuencias. Este enfoque es de gran relevancia, hace posible obtener una representación más compleja en comparación con los métodos tradicionales, permitiendo así una inferencia más precisa y eficiente (LeCun, Bengio, y Hinton, 2015).

Diversos autores han propuesto una variedad de enfoques y criterios para analizar y detectar actividades mediante el uso de clasificadores de imágenes. Estas técnicas se basan en las capacidades de los algoritmos de clasificación para identificar patrones de movimiento y asociarlos con distintas actividades. No obstante, es crucial destacar que la complejidad computacional en estos enfoques teóricos puede limitar su aplicabilidad en contextos más amplios (Sun y cols., 2023; Gu y cols., 2021).

Sin embargo, estos la mayoría de los enfoques y métodos son limitados en su capacidad y acceso a la población para interpretar actividades humanas complejas debido a su enfoque en características de bajo nivel y su susceptibilidad a las variaciones en las condiciones de captura, como la iluminación y el ángulo de la cámara. Las acciones humanas que aparecen en los sistemas de vigilancia pueden ser afectadas según los movimientos, velocidad, tipo de cámara, apariencia y variaciones de pose, etc., haciendo que la representación de la acción sea una actividad compleja (Poppe, 2010; Kulsoom y cols., 2022).

Es por ello por lo que se han desarrollado diversos enfoques para mejorar la precisión y eficiencia de la detección de actividades. Seong-Wook Joo y R. Chellappa propusieron una gramática para reconocer actividades usando etiquetas y reglas sintácticas para describir eventos (Kumar y Kumar, 2024). Este método permite una descripción estructurada de las acciones, facilitando su identificación en secuencias de vídeo (Pirsiavash y Ramanan, 2014).

La mayoría de estos enfoques operan en sistemas centralizados, donde toda la información y el procesamiento se concentran en un punto central o servidor. Sin embargo, los sistemas distribuidos representan una alternativa donde estas funciones se descentralizan y un conjunto de componentes autónomos, los cuales se perciben como un sistema único y coherente para los usuarios y las aplicaciones. Los sistemas distribuidos distribuyen las tareas y los datos en múltiples nodos, proporcionando redundancia, escalabilidad y eficiencia (Karim y cols., 2024).

Los sistemas distribuidos están orientados para resolver problemas exten-

sos, debido a la cantidad de información y rendimiento que poseen, además de la flexibilidad, escalabilidad, tolerancia a fallos y compartir recursos (Steen y Tanenbaum, 2016; Abad y cols., 2021).

Los lenguajes diseñados para sistemas distribuidos, como Argus, Emerald, Linda, Bloom y Lasp, ofrecen ventajas específicas para la gestión de actividades en entornos distribuidos (Liskov, 1988; Almes, Black, Lazowska, y Noe, 1985; Black, Hutchinson, Jul, Levy, y Carter, 1987). La mayoría de ellos están orientados a facilitar la comunicación y coordinación entre los componentes del sistema, así como a manejar la configuración de la red. Además, existen herramientas y bibliotecas complementarias, como RMI (Remote Method Invocation) y CORBA (Common Object Request Broker Architecture), que permiten configurar y gestionar la interacción entre objetos distribuidos en diferentes entornos de ejecución.

Argus, se destaca por su capacidad de manejar fallos y mantener la consistencia a través de mecanismos de recuperación y persistencia de datos. Emerald, por otro lado, facilita la migración de objetos en un sistema distribuido, permitiendo una mayor flexibilidad y eficiencia (Liskov, 1988). Linda se enfoca en la comunicación y coordinación mediante un espacio de tuplas, lo que simplifica la programación distribuida y la sincronización de procesos. Bloom introduce un modelo declarativo basado en lógica, lo que permite expresar programas distribuidos de manera concisa y comprensible (Almes y cols., 1985; Black y cols., 1987).

Finalmente, Lasp se especializa en la programación reactiva y el procesamiento de flujos de datos en tiempo real, siendo ideal para aplicaciones que requieren una rápida adaptación a cambios en el entorno. En conjunto, estos lenguajes ofrecen diversas estrategias y herramientas para superar los desafíos inherentes a los sistemas distribuidos, mejorando la eficiencia, flexibilidad y capacidad de respuesta en la detección e inferencia de actividades (Meiklejohn y Van Roy, 2015).

Además de los lenguajes previamente mencionados, otros lenguajes relacionados con la descripción de actividades en sistemas distribuidos se destacan por

sus enfoques específicos y aplicaciones particulares. Entre estos, se encuentran ECA (Event-Condition-Action), ADeL (Activity Description Language) e ILIAD (Interactive Learning from Activity Description) (Alferes, Banti, y Brogi, 2006; Sarray, Ressouche, Moisan, Rigault, y Gaffe, 2017; Nguyen, Phung, Venkatesh, y Bui, 2005).

ECA establece una gramática de reglas que asigna un significado a acciones basadas en condiciones específicas durante eventos particulares. ADeL introduce un modelo jerárquico de actividades utilizando autómatas finitos, integrando roles, eventos y subactividades para una definición natural de las actividades. ILIAD, por otro lado, proporciona un protocolo interactivo de aprendizaje que permite a los agentes recibir instrucciones verbales y describir sus acciones para el entrenamiento.

ICA y ADeL son lenguajes adecuados para la configuración de redes de vigilancia y control de actividades predefinidas, pero pueden enfrentar dificultades para escalar o adaptarse a escenarios dinámicos.

Estos lenguajes descritos trabajan bajo la centralización del sistema; sin embargo, la utilización de lenguajes distribuidos ha tenido diferentes objetivos. Principalmente, están enfocados a configurar componentes y generar comunicación en un sistema distribuido. Erlang, Eden, Ada, Scala, Akka, Haskell y otros son lenguajes que están enfocados a configurar elementos de la red o de la comunicación entre los componentes (Rakhimov y cols., 2020).

Finalmente, Dias propone VigiLANG, un lenguaje de alto nivel diseñado para sistemas de vigilancia distribuidos, basado en COACH UNILANG y ECOLANG, facilitando la comunicación entre componentes autónomos como cámaras y un agente central en el sistema de vigilancia (Dias, Rocha, Silva, Leao, y Reis, 2005a). VigiLANG, Erlang y Scala están diseñados para operar en entornos distribuidos donde múltiples componentes deben comunicarse de manera eficiente y coordinada (Alferes y cols., 2006; Sarray y cols., 2017; Nguyen y cols., 2005; Jensen, Kristensen, y Wells, 2007; Vella y cols., 2021; Zerzour y Frazier, 2000).

Estos lenguajes permiten una mayor flexibilidad y adaptabilidad al modelar

e inferir actividades en escenarios complejos y dinámicos. Facilitando la interoperabilidad entre sistemas y la gestión descentralizada de recursos, lo cual es crucial para aplicaciones de sistemas de vigilancia (Alevizos y cols., 2017).

Estos lenguajes son herramientas para la descripción y control de actividades en sistemas de vigilancia centralizados. Sin embargo, estos lenguajes aún están limitados para adaptarlos al modelado e inferencia de actividades complejas a través de diversos escenarios.

Estos lenguajes permiten una mayor flexibilidad y adaptabilidad al modelar e inferir actividades en escenarios complejos y dinámicos. Facilitando la interoperabilidad entre sistemas y la gestión descentralizada de recursos, lo cual es crucial para aplicaciones de sistemas de vigilancia (Dragon, Fenzi, Siberski, Rosenhahn, y Ostermann, 2012).

Los diferentes lenguajes distribuidos muestran una amplia diversidad de características y plataformas diseñadas para facilitar el desarrollo de sistemas distribuidos, cada uno orientado a necesidades (Dias, Rocha, Silva, Leao, y Reis, 2005b). El uso de lenguajes distribuidos representa un avance significativo en la productividad y efectividad de sistemas en vigilancia mejorando la adaptabilidad, flexibilidad en los sistemas.

Capítulo 3

Metodología

En los capítulos anteriores, se presentó la información importante para la definición del lenguaje distribuido, las características de los sistemas distribuidos, así como algunos antecedentes relevantes. En este capítulo, se expone la propuesta para modelar comportamientos utilizando un lenguaje formal distribuido basado en primitivas de movimiento, describiendo cada uno de los elementos que componen el sistema.

El sistema distribuido, con sus diversos componentes, tiene como función principal mantener la heterogeneidad del sistema, asegurando así su transparencia para el usuario. Además, ofrece una variedad de servicios, entre los que se incluyen el análisis de la información y la comunicación eficiente entre los componentes, lo que mejora la funcionalidad y la integración del sistema en su conjunto (Emmerich, Aoyama, y Sventek, 2008).

El sistema propuesto pone énfasis principalmente en el paralelismo, la comunicación, el desempeño y la tolerancia a fallos, además de la extensibilidad, modularidad, portabilidad, escalabilidad y otras características que aportan los sistemas distribuidos (Suljkanović, Gostojić, Dejanović, y Kaplar, 2017).

La metodología de este trabajo se estructuró en torno a los objetivos particulares definidos, siguiendo una serie de pasos clave descritos a continuación.

Primero, se definieron las primitivas de movimiento necesarias para modelar los eventos en sistemas multicámara. Estas primitivas constituyen las unidades básicas de acción que el sistema utiliza para analizar y modelar comportamientos. Después, se estableció la gramática del lenguaje distribuido, un componente crucial que define las reglas para interpretar y procesar las acciones dentro del sistema.

Este lenguaje organizará la información de manera coherente, permitiendo al sistema describir e inferir comportamientos complejos. Actúa como un marco estructural que organizará la información y permitirá al sistema comprender de manera uniforme las actividades captadas por las cámaras.

El siguiente paso es el diseño de la arquitectura de un sistema distribuido capaz de procesar datos provenientes de múltiples cámaras. Este diseño asegura la escalabilidad, flexibilidad y capacidad de respuesta del sistema, permitiendo la interconexión eficiente de sus diversos componentes.

Una vez establecida la arquitectura, se trabajará en el diseño del sistema de comunicación entre los componentes del sistema distribuido. Este paso garantizará que las cámaras y las unidades de procesamiento intercambien información de manera efectiva, logrando una integración sólida y un flujo continuo de datos en tiempo real.

Con el sistema distribuido en funcionamiento, se desarrolló un conjunto de programas en el lenguaje definido. Estos programas permiten identificar y modelar actividades en diferentes escenarios multicámara, asegurando que las reglas establecidas por la gramática sean aplicadas de manera eficiente para describir eventos.

Finalmente, se valida la experimentación del sistema en escenarios reales. Durante esta etapa, se evalúa la capacidad del sistema para modelar e identificar actividades con precisión. Los resultados obtenidos proporcionan una medida clara de su efectividad y servirán como base para realizar ajustes y optimizaciones adicionales.

En conjunto, esta metodología establece un marco sólido para la detección y análisis de actividades en sistemas distribuidos multicámara. En la Figura 3.1, se presenta una representación visual que ilustra detalladamente cada uno de los pasos descritos, desde la definición de las primitivas de movimiento hasta la validación experimental.

Además, se destacó que el enfoque distribuido del sistema facilita la escalabilidad y asegura que se puedan incorporar nuevas cámaras o módulos de análi-

sis sin afectar el rendimiento general. Este diseño modular y extensible sienta las bases para aplicaciones en entornos dinámicos y complejos.

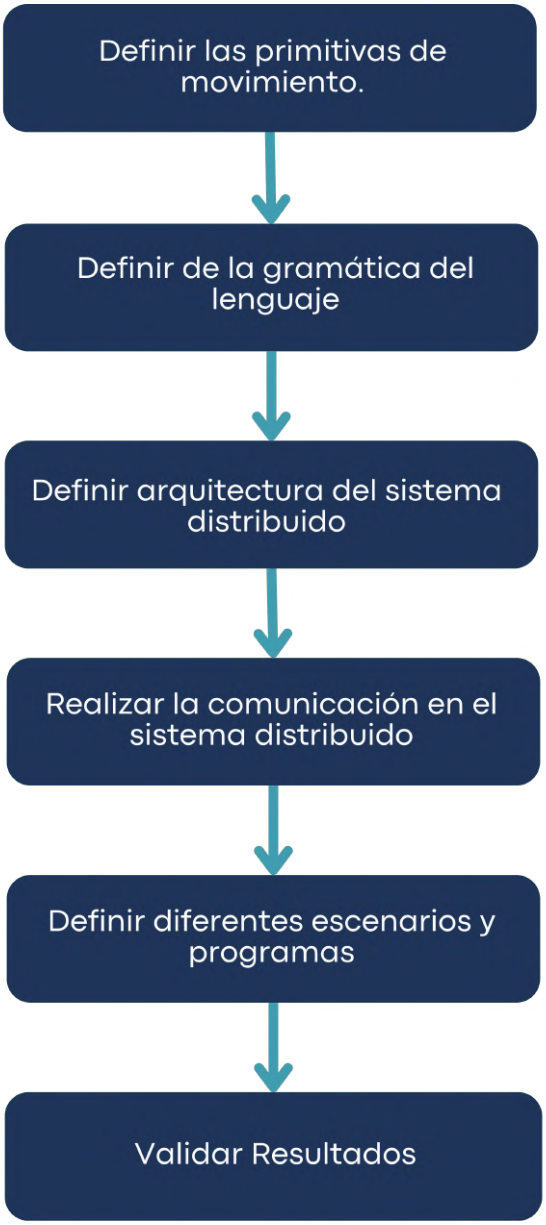


Figura 3.1: Metodología propuesta.

3.1 Lenguaje SEL

3.1.1. Escenarios de sistemas multicámara

Un sistema distribuido en términos de sistemas de vigilancia en aquellos que son multicámara, donde múltiples cámaras, ubicadas en diferentes posiciones físicas, trabajan de forma coordinada para analizar escenas desde distintos ángulos y generar una visión unificada del entorno observado(Abad y cols., 2021).

La representación de los escenarios es basada en los componentes que se agreguen al sistema multicámara, teniendo así un conjunto de cámaras o sensores que nos dan una representación de un entorno.

En el sistema distribuido multicámara se tienen cierto tipo de escenarios, estos pueden ser área no solapada, área solapada y área discontinua. Cada una de estas configuraciones es importante analizarlas debido a que nos puede ayudar en el modelado e inferencia de las actividades, en la Figura 3.2 se puede ver un ejemplo de cada uno de los tipos de escenarios que se pueden presentar en el sistema multicámara.

- **Área solapada**: Este escenario se caracteriza por tener áreas que se comparten parcialmente entre las vistas en los escenarios.
- **Área no solapados**: En este escenario los nodos no comparten áreas en común.
- **Área Discontinua**: Este escenario se caracteriza por tener áreas separadas físicamente lo cual puede generar áreas huecas sin analizar.

Cada uno de los escenarios definidos previamente cuenta con ciertas características, los escenarios no solapados tienen limitaciones en términos de detección de objetos que se mueven entre diferentes áreas, ya que no existe una cobertura continua, en los escenarios solapados suelen ser escenarios que

generan redundancia en los datos y por último la principal limitante en los escenarios discontinuos es falta de visibilidad sobre las zonas no cubiertas lo que puede dificultar el análisis global del entorno.

Después de conocer y analizar el tipo de escenarios que se cuenta en el entorno distribuido, cada uno de estos escenarios tendrán que segmentarse de manera individual a través de estados.

Los estados son unidades fundamentales que representan regiones espaciales de un solo escenario que son definidas por el usuario. Cada estado genera una lista distinta de posiciones en la escena, basada en la matriz especificada. Estas regiones se definen para verificar la ausencia o presencia de movimiento (Ramírez, García, y Jiménez, 2018).

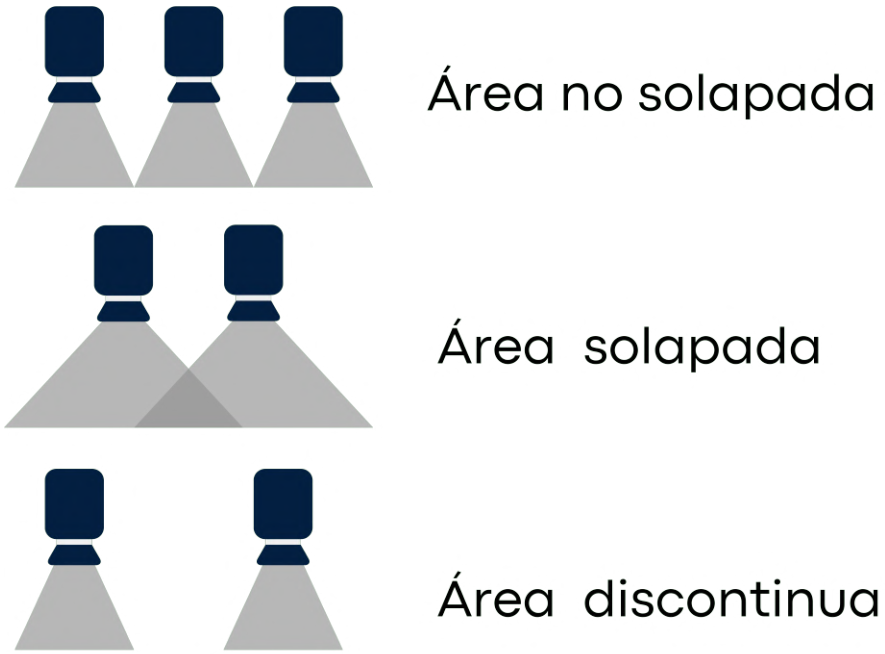


Figura 3.2: Tipos de escenarios en un sistema multicámara.

Los estados definidos pueden presentarse en dos formas: activos/inactivos (presencia/ausencia de movimiento). La activación de un conjunto particular representa un evento, y cada vez que se vuelve *verdadero*, significa que se ha detectado movimiento en los píxeles que conforman el estado. El usuario tiene la flexibilidad de segmentar los estados y definir la información de acuerdo con su preferencia. Sin embargo, esta tarea puede ser desafiante, ya que requiere un

análisis detallado de los escenarios y sus relaciones entre ellos.

El usuario define las divisiones de estados mencionadas, las cuales pueden aplicarse en el contexto de la Figura 3.3. Esta figura ilustra el proceso de segmentación y etiquetado basado en un escenario $I(\mathbf{x})$, produciendo finalmente una lista de posiciones de estados en la imagen.

Estos estados son determinados a través del análisis de cada uno de los escenarios, en la Figura 3.3 se observa un escenario que es segmentado en tres estados, la segmentación se hace de acuerdo con ciertas características del escenario o de las actividades que se quieren modelar.

3.1.2. Primitivas de movimiento

SEL introduce un enfoque para caracterizar actividades mediante relaciones temporales entre estados, utilizando primitivas de movimiento como secuencia, paralelismo y concurrencia.

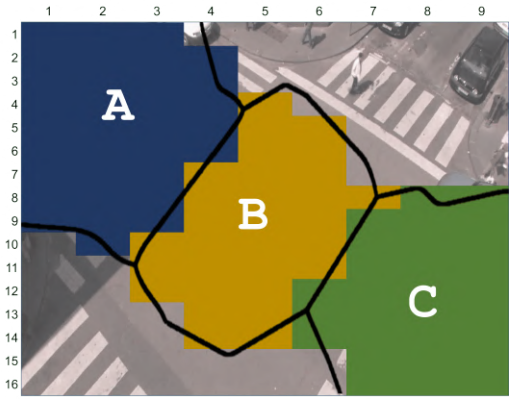
Estas primitivas están diseñadas para describir cómo evolucionan las actividades a lo largo de trayectorias, la simultaneidad de eventos y la sincronización de movimientos representados en los estados de cada escenario, siendo fundamentales para representar el movimiento dentro del sistema propuesto.

La incorporación de estos operadores tiene como objetivo mejorar la capacidad del sistema para expresar de manera precisa y detallada estructuras más complejas en el marco del lenguaje formal.

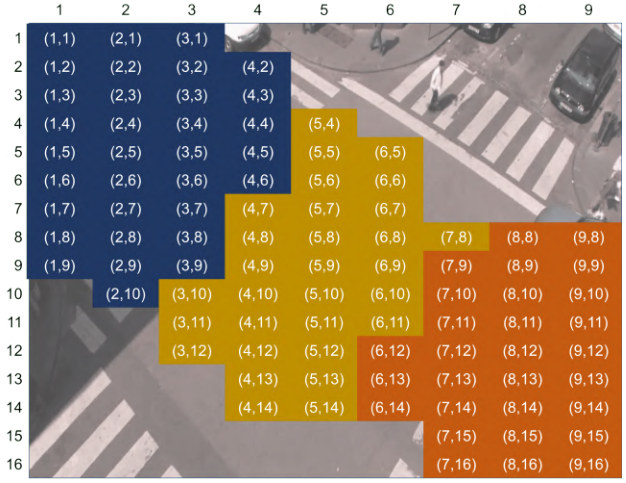
Las relaciones temporales se definen como un flujo unidireccional con incremento constante (frecuencia de adquisición), lo que permite evaluar la presencia o ausencia de movimiento en cada estado. Esta representación permite visualizar las secuencias de video como imágenes que evolucionan a lo largo del tiempo.

El *operador de secuencia* (seq) para algunos estados dado $p \in P$ y $q \in Q$, ya que pertenecen a espacios diferentes, se define como $seq : P \times Q \rightarrow \{ true, \ false\}$; es decir, toma un conjunto de estados respectivamente de uno o más escenarios y verifica que el siguiente estado en la lista se active después

(a) Escenario original.



(b) Escenario segmentado



(c) Posiciones en la matriz

```
state street.A=[(1,1),(1,2),(1,3),(2,1),(2,2),(2,3),(2,4),(3,1),(3,2),
               (3,3),(3,4),(4,1),(4,2),(4,3),(4,4),(5,1),(5,2),(5,3),
               (5,4),(6,1),(6,2),(6,3),(6,4),(7,1),(7,2),(7,3),(8,1),
               (8,2),(8,3),(9,1),(9,2),(9,3),(10,2)];
state street.B=[(4,5),(5,5),(5,6),(6,5),(6,6),(7,4),(7,5),(7,6),
               (8,4),(8,5),(8,6),(8,7),(9,4),(9,5),(9,6),(10,3),
               (10,4),(10,5),(10,6),(11,3),(11,4),(11,5),(11,6),
               (12,3),(12,4),(12,5),(13,4),(13,5),(14,4),(14,5)];
state street.C=[(8,8),(8,9),(9,7),(9,8),(9,9),(10,7),(10,8),(10,9),
               (11,7),(11,8),(11,9),(12,6),(12,7),(12,8),(12,9),
               (13,6),(13,7),(13,8),(13,9),(14,6),(14,7),(14,8),
               (14,9),(15,7),(15,8),(15,9),(16,7),(16,8),(16,9)];
```

(d) Código

Figura 3.3: Segmentación de estados en un escenario.

de que el estado anterior se haya completado, continuando de esta forma hasta finalizar.

$$seq(\mathbf{P}.p, \mathbf{Q}.q) \iff \mathbf{P}\,(p; i, j) \wedge \mathbf{Q}\,(q; j+1, k) \wedge \left(\overrightarrow{q}^{j+1} =\right) \tag{3.1}$$

Para una marca de tiempo arbitraria i, j, k con la restricción $i < j < k$. El término $\overrightarrow{a}^{j+1} =$ false denota la restricción de orden temporal.

El *operador paralelo* se refiere a la activación concurrente de dos estados según el escenario. Para un intervalo de tiempo dado $[i, j]$, donde a ejecución no necesariamente es mismo tiempo, pueden intercalarse en tiempo, o presentarse de manera sincronizada.

$$par(\mathbf{P}.p, \mathbf{Q}.q) \iff (\mathbf{P}(p; i, j) \vee \mathbf{Q}(q; i, j)) \vee (\mathbf{P}(p; i, j) \wedge \mathbf{Q}(q; i, j)) \tag{3.2}$$

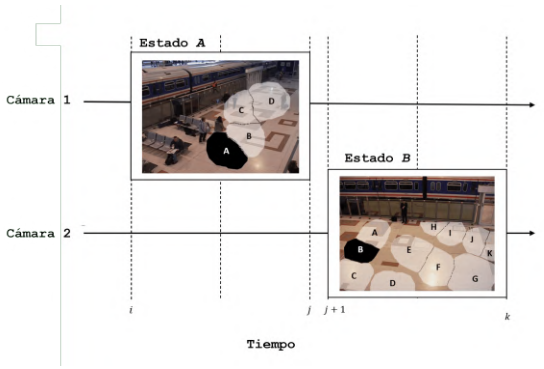
tal que par denota el símbolo del operador, que se define como $par : P \times Q \rightarrow \{\text{true}, \text{false}\}$. Como se aprecia en un momento determinado, dos estados se vuelven en paralelo si uno de ellos, o ambos, presentan movimientos simultáneamente.

El *operador concurrente* se refiere a la ejecución sincronizada de dos o mas estados con la restricción de que ambos se encuentran activos hasta que alguno de se desactiva, el *operador concurrente* se define como sigue.

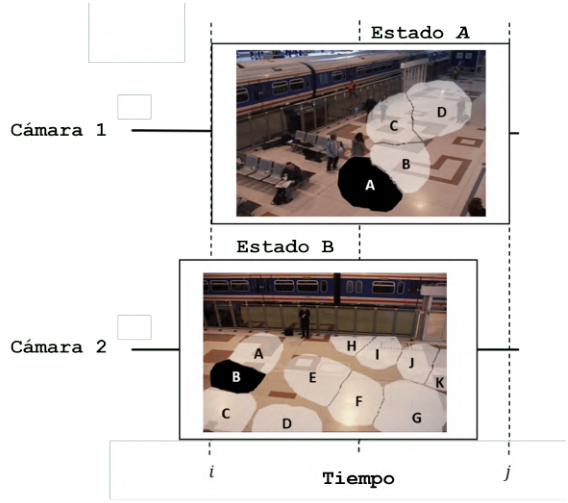
$$con(\mathbf{P}.p, \mathbf{Q}.q) \iff \mathbf{P}(q; i, j) \wedge \mathbf{Q}(q; i, j) \tag{3.3}$$

tal que con denota el símbolo del operador, que se define como $con : P \times Q \rightarrow \{\text{true}\}$.

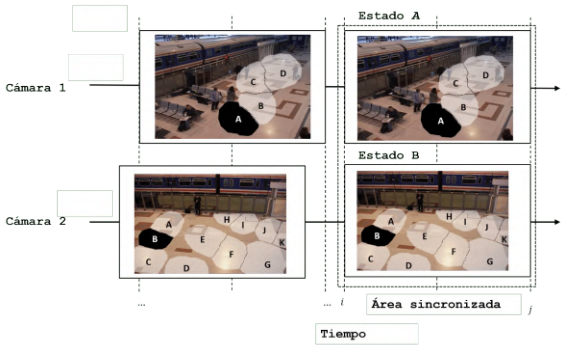
En conclusión, el operador secuencia describe trayectorias, el operador paralelismo representa actividades que ocurren simultáneamente, y el operador concurrencia se refiere a la sincronización de eventos. El uso de estas primitivas nos permite construir actividades de manera recursiva, lo que facilita modelar la dinámica de los objetos en múltiples escenarios con características diversas.



(a) Secuencia



(b) Paralelismo



(c) Concurrencia

Figura 3.4: Primitivas de movimiento en diferentes escenarios.

3.1.3. Definición de la gramática

En la práctica, cuando una cadena o palabra es validada por un lenguaje a través de un proceso de aceptación, este analiza las transiciones de estado y las reglas internas que coinciden con las gramáticas y reglas del lenguaje.

Las primitivas de movimiento descritas previamente se representan como comandos dentro del lenguaje, permitiendo que cualquier programa codificado con esta gramática incorpore las primitivas de manera recursiva. Estos operadores están asociados a las actividades de movimiento capturadas por una cámara.

La Figura 3.5 presenta la gramática del lenguaje, detallando las definiciones de las reglas gramaticales relacionadas con el modelo propuesto. También se describen los identificadores, palabras clave y símbolos, así como la estructura sintáctica requerida para programar en este lenguaje.

```
[program]::=[header][states][body]
[header]::= module [id];
[states]::= [state]| [state] [states]
[sensor]::=[id].[id]
[state]::= state [sensor] = [[area]];
[body]:: begin; [statements] end;
[activities]::= [activity] | [activity] [activities]
[activity]::= [primitive]([sensor],[sensor]|[sensor]|null); [activityname]
[activity]::= [primitive]([sensor], [sensor] | [sensor], null); [activityname]
[area]::= ([number],[number]), [area] | ([number],[number])
[activityname]::= %[id];
[primitive]::= seq | par | con
[id]::= [letter] [string]
[string]::= <letter> <string> | <number> <string> | ε
[letter]::= [A-Za-z]
[number]::= [0-9]
```

Figura 3.5: Definición básica y reglas gramaticales para el lenguaje formal propuesto.

El lenguaje SEL(Semantic Event Language) es basado en la descripción de eventos multicámara los cuales hacen referencia a cada componente disponible en el sistema distribuido. Los escenarios se definen por medio de variables. A la vez, cada escenario tiene sus estados correspondientes, de acuerdo con las necesidades de las actividades que se desean modelar.

El lenguaje SEL se basa en la descripción de eventos multicámara, los cuales hacen referencia a cada componente disponible en el sistema distribuido. Los

escenarios se definen mediante variables, y cada uno de ellos cuenta con variables internas que representan áreas bien delimitadas dentro de la imagen de cada componente. Estas áreas, denominadas estados, se establecen según las necesidades específicas de las actividades que se desean modelar.

Los programas que utilizan la gramática SEL funcionan como entrada para un nuevo proceso dentro del sistema de vigilancia distribuido. Cada programa debe contar con una estructura dividida en tres secciones: $Header$, $States$ y $Body$. En la Figura 3.6 se presenta un ejemplo de un programa en SEL. La sección $Header$ contiene el nombre del programa, mientras que la sección $States$ define la segmentación de cada uno de los componentes registrados en el sistema multicámara distribuido.

Figura 3.6: Ejemplo de las partes del lenguaje SEL.

Cada estado en los escenarios produce una única lista de posiciones dentro de la escena, acompañada de una etiqueta o variable definida. El estado representa el área espacial utilizada para probar la existencia o ausencia de movimiento.

Finalmente, en la sección $Body$, se define o modela las actividades a través de las primitivas de movimiento. Los estados son la unidad básica del lenguaje y

permiten la creación de sentencias para definir actividades mediante las tres primitivas: secuencia, paralelismo y concurrencia, las cuales describen la actividad en función de cómo se comportan los estados a lo largo del tiempo.

Cada uno de los diferentes tipos de escenarios deben de considerarse al modelar las diversas actividades, así como la segmentación de cada escenario.

Ahora cuando se tiene un escenario de tipo de área discontinua nos enfrentamos a una mayor incertidumbre principiante al modelar actividades como una trayectoria (secuencia), ya que en un lapso de tiempo/área no se presentara movimiento por que no se tiene acceso a esa área, por lo cual se utilizara la palabra $null$ la cual gramaticalmente significa ausencia de movimiento por un determinado tiempo/área, es decir en el caso de modelar actividades y presentar esa discontinuidad de área, será que se presente la palabra $null$.

Este lenguaje asume que una actividad puede describirse como una secuencia de estados que se activan a lo largo del tiempo mediante las primitivas de movimiento. La gramática propuesta se estructura en una segmentación definida por el usuario, junto con primitivas de movimiento que determinan el comportamiento y la dinámica de la actividad.

Cada uno de los diferentes tipos de escenarios debe ser considerado al modelar las diversas actividades, así como la segmentación de cada escenario.

Cuando se trabaja con un escenario de tipo área discontinua, se presenta una mayor incertidumbre al modelar actividades como trayectorias (secuencias). Esto se debe a que, durante ciertos intervalos de tiempo o en determinadas áreas, no se detectará movimiento debido a la inaccesibilidad de esas zonas. En estos casos, se utilizará el término $null$, que gramaticalmente denota la ausencia de movimiento en un área o período específico. Así, al modelar actividades y representar esta discontinuidad en el área, el uso de $null$ indica la falta de movimiento en las regiones no accesibles.

Una de las características principales de la gramática es su capacidad para emplear las cámaras definidas en la descripción de los estados, permitiendo su referencia a través de una notación basada en puntos. Esta notación facilita la

invocación de los estados de cada componente del sistema de forma organizada y precisa. Al utilizarla, los estados se pueden definir y referenciar de manera específica para cada cámara, lo que optimiza la gestión del modelado de las actividades y el control del sistema distribuido multicámara.

Este lenguaje asume que una actividad puede describirse como un conjunto de estados que se activan a lo largo del tiempo utilizando las primitivas de movimiento. La gramática propuesta se compone de una segmentación definida por el usuario y primitivas de movimiento que determinan la actividad.

Cada programa es definido por el usuario, quien determina las actividades mediante las primitivas de movimiento, los estados y los operadores. El intérprete realiza un análisis sintáctico, léxico y semántico de cada uno de los programas, con el objetivo de traducir las instrucciones y permitir la comunicación de la información con el sistema distribuido multicámara.

3.2 Arquitectura de un Sistema Distribuido

En la literatura, se emplean diversos términos para describir los sistemas distribuidos. Una de las definiciones más acertadas podría ser la de un conjunto de computadoras que son autónomas y físicamente separadas, pero son percibidas por el usuario como un sistema integrado y único (Abad y cols., 2021). Los sistemas distribuidos están compuestos por múltiples componentes que se comunican entre sí para realizar tareas específicas, ocultando cierta información al usuario y colaborando para ejecutar dichas tareas a través de esta comunicación. El uso de sistemas distribuidos está orientado a resolver problemas complejos, gracias a su capacidad para manejar grandes volúmenes de información y su alto rendimiento. Además, ofrecen flexibilidad, escalabilidad y tolerancia a fallos (Van Steen y Tanenbaum, 2016).

La arquitectura de un sistema distribuido se refiere a la forma en que se estructura y se comunican los diferentes componentes del sistema. Definir la arquitectura es crucial para el diseño de un sistema distribuido, ya que permite entender y organizar cómo se ejecutará el sistema (van Steen y Tanenbaum,

2023). Además, este proceso permite identificar a las partes interesadas clave y contribuye a reducir las brechas de comunicación (Klein y van Vliet, 2013).

Diversos estilos arquitectónicos determinan la organización fundamental de un sistema. Estos modelos constituyen uno de los principales desafíos en la computación distribuida, ya que deben garantizar la correcta conexión entre componentes a través de una red de datos. Algunos ejemplos comunes de estilos arquitectónicos utilizados en la computación distribuida son: el modelo de capas, cliente/servidor, P2P (peer-to-peer), y la arquitectura orientada a servicios (SOA), entre otros (Abad y cols., 2021).

Cada uno de estos modelos arquitectónicos ofrece estructuras y patrones específicos que optimizan la distribución, la interacción y el rendimiento de los sistemas distribuidos.

Los sistemas distribuidos ofrecen varias ventajas, existen factores importantes que pueden afectar su funcionalidad, tales como la falta de estándares y soporte, la complejidad y la seguridad de los datos (Abad y cols., 2021). Por lo tanto, la propuesta incorpora una arquitectura de sistema distribuido adecuada para la implementación del lenguaje propuesto, el cual permite representar, inferir y diseñar operaciones en el área visual a través de un sistema distribuido.

La propuesta se basa en un sistema de vigilancia distribuido multicámara, cuyo objetivo principal es abordar la heterogeneidad, garantizando transparencia para el usuario. Además, proporciona una variedad de servicios, que incluyen el análisis de información y la comunicación entre los componentes, mediante un lenguaje diseñado para la configuración del sistema distribuido y para el modelado de actividades en distintos escenarios.

La arquitectura de cualquier sistema es fundamental para definir los componentes, identificar las partes interesadas y establecer la comunicación entre ellos (Klein y van Vliet, 2013).

El estilo arquitectónico seleccionado es de capas, debido a su capacidad para estructurar la aplicación en niveles bien definidos. Cada capa representa un nivel de abstracción que se comunica exclusivamente con las capas adyacen-

tes, siguiendo el principio de separación de responsabilidades. Esta organización favorece la modularidad y la reutilización de componentes, así como la escalabilidad del sistema, ya que permite la adición o modificación de capas sin afectar a las demás (Contreras Rivas y cols., 2025). Además, el estilo de capas promueve la interoperabilidad y la independencia tecnológica entre las distintas capas, lo cual es crucial en entornos distribuidos donde la heterogeneidad de plataformas y servicios es común. Esta independencia facilita la implementación de cambios, así como la asignación o eliminación de componentes de manera eficiente (Vatter, Mayer, y Jacobsen, 2023).

Una de las principales ventajas de la arquitectura en capas es su modularidad. Cada capa funciona como una unidad independiente, lo que facilita la adición, modificación o eliminación de capas sin afectar las demás (Abad y cols., 2021). La separación de responsabilidades entre capas también simplifica el mantenimiento del sistema, reduciendo el riesgo de introducir nuevos errores durante el proceso de corrección. Además, la arquitectura en capas mejora la escalabilidad del sistema.

Finalmente, la arquitectura en capas aborda eficientemente los requisitos de abstracción, rendimiento y estandarización en sistemas de visión. Este enfoque es escalable y portable, lo que permite adaptarse a diferentes entornos y necesidades de manera efectiva.

En la Figura 3.7 se muestra la arquitectura en capas del sistema propuesto, que consta de tres capas independientes: aplicación, control y propiedades. Adoptar una arquitectura en capas es crucial para mantener un control eficiente y proporcionar una variedad de servicios, incluidos el análisis de información y la comunicación entre componentes, de manera transparente y amigable para el usuario.

La capa de aplicación es la interfaz directa con el usuario final y se encarga de proporcionar una experiencia clara e intuitiva. La capa de aplicación se comunica con la capa de control para obtener información sobre los componentes registrados en la capa de propiedades. A través de la capa de aplicación,

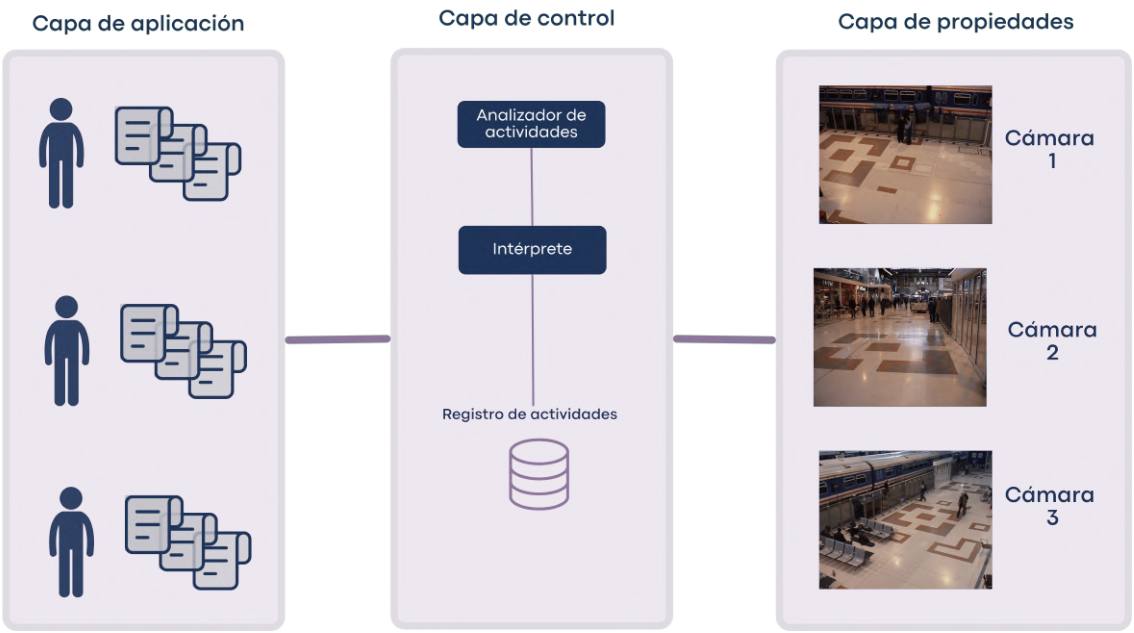


Figura 3.7: Arquitectura de capas en vista física.

los usuarios pueden acceder a información relevante de cada componente del sistema, visualizar escenarios específicos y segmentar áreas de interés.

Además, ofrece herramientas y funcionalidades que permiten a los usuarios interactuar con el sistema. Esta capa también facilita la integración de diferentes dispositivos y servicios, proporcionando una plataforma unificada desde la cual los usuarios pueden gestionar, modelar e inferir actividades en el sistema de vigilancia.

Por otra parte, la capa de control asume la responsabilidad principal del sistema, ya que se encarga del análisis, interpretación y traducción de la información basada en eventos, siendo el núcleo lógico que coordina el funcionamiento general del sistema.

Esta capa no solo gestiona la comunicación entre todos los componentes del sistema, sino que también valida y administra la información. Todo esto se realiza de manera transparente y sencilla para el usuario, garantizando interacciones fluidas y eficientes.

La capa de control asegura que los datos se procesen correctamente y que las respuestas a las solicitudes del usuario sean precisas y oportunas, facilitando

una experiencia óptima y un funcionamiento eficaz del sistema en su conjunto (Cob-Parro, Losada-Gutiérrez, Marrón-Romera, Gardel-Vicente, y Bravo-Muñoz, 2021; Chaudhary y cols., 2021).

En la capa de control se reciben todas las peticiones del usuario y se envían respuestas tras realizar el procesamiento correspondiente. Esta capa es responsable de coordinar y gestionar todas las operaciones solicitadas, asegurando que cada solicitud se procese adecuadamente antes de enviar la respuesta al usuario. Su función es crucial, ya que actúa como intermediario entre el usuario y las diferentes capas del sistema, garantizando un flujo de información ordenado y eficiente.

En la capa de control se reciben todas las peticiones del usuario y se envían respuestas tras realizar el procesamiento correspondiente. Esta capa es responsable de coordinar y gestionar todos los procesos del sistema. Además, tiene la responsabilidad de validar las entradas y salidas de los datos (Cob-Parro y cols., 2021; Chaudhary y cols., 2021).

La capa de control está compuesta por un directorio de direcciones, que actúa como una base de datos donde están registrados los componentes del sistema de vigilancia, como cámaras o sensores, que residen en la capa de propiedades. Esta capa de control puede acceder a la información de cada componente para analizar las actividades a través de llamadas (Abad y cols., 2021).

Cuando la información sobre el movimiento llega a la capa de control se encarga de analizarla. Si el patrón detectado coincide con alguna de las actividades previamente definidas por el usuario, se envía una alerta informando que la actividad ha sido completada. Para ello, los componentes del sistema como cámaras o sensores son quienes capturan y procesan las imágenes, con el objetivo de detectar movimiento en los estados. Esta información es posteriormente enviada a un agente central, encargado de consolidar y evaluar los datos para la toma de decisiones.

Finalmente, la capa de propiedades se encarga de gestionar los componentes y el origen de los datos, facilitando el acceso o la recuperación de información

proveniente de múltiples componentes como cámaras o sensores (Tappenden, Huynh, Miller, Geras, y Smith, 2006).

Esta capa se encarga de acceder, recuperar y organizar la información capturada por los dispositivos, garantizando que los datos estén disponibles para su procesamiento posterior. Además, la capa de propiedades facilita la integración de diversos tipos de cámaras o sensores, promoviendo la interoperabilidad y la comunicación entre ellos, sin importar sus especificaciones técnicas o plataformas.

Al gestionar eficientemente el flujo de datos desde su origen, esta capa asegura que la información relevante sea transmitida de manera oportuna y precisa a las capas superiores para su análisis y toma de decisiones (Tappenden y cols., 2006). A pesar de que los escenarios pueden variar significativamente, esta capa mantiene una comunicación efectiva dentro del sistema de vigilancia distribuido mediante las solicitudes de servicios realizadas en la capa de control.

Los componentes en la capa de propiedades actúan como agentes autónomos que obtienen la información a analizar, ya sea localmente o a través de diversos métodos de detección de movimiento, intercambiando información relevante según el escenario. Estos componentes pueden estar desarrollados en diferentes lenguajes de programación y ejecutarse en diversas plataformas y sensores. A pesar de que los escenarios pueden ser completamente distintos, mantienen una comunicación efectiva dentro del sistema de vigilancia distribuido a través de las solicitudes de servicios en la capa de control.

Los escenarios ahora están interconectados a través de los diferentes tipos de componentes, cada uno de los cuales aporta características específicas orientadas a beneficios y objetivos definidos. Es crucial mantener la comunicación entre los componentes, ya que esto permite un mayor acceso a la información y proporciona un mejor conocimiento para analizar y predecir eventos (Berger y Lu, 2022).

3.2.1. Detección de Movimiento

En la literatura, existen diversas técnicas para la detección de movimiento en imágenes. Entre las más destacadas, se encuentran aquellas que utilizan la información espacial y temporal de la secuencia de imágenes, como la sustracción de fondo, las diferencias temporales y el flujo óptico. Cada una de estas técnicas presenta características y objetivos particulares, como la luminosidad, los reflejos, las sombras, la velocidad, el costo computacional, las condiciones climáticas, las oclusiones, entre otros (Morris y Trivedi, 2008).

Existen dos enfoques principales para la detección de movimiento o identificación de objetos en un escenario: diferencia temporal y sustracción de fondo. La diferencia temporal consiste en restar dos fotogramas consecutivos y aplicar un umbral, mientras que la sustracción de fondo se basa en restar un fondo o modelo de referencia de la imagen actual. En ambos métodos, se pueden usar operaciones morfológicas para reducir el ruido de los resultados obtenidos (Valera y Velastín, 2005).

La técnica de diferencias temporales está principalmente orientada a escenarios estáticos, ya que se enfoca en la comparación entre fotogramas consecutivos para identificar cambios. Por otro lado, la sustracción de fondo es una técnica ampliamente utilizada para detectar objetos en movimiento a partir de cámaras estáticas y un fondo conocido. Esta técnica permite distinguir rápidamente eventos en movimiento y, además, facilita la realización de estudios sobre la movilidad de los objetos (Sobral y Vacavant, 2014; Richard y Gall, 2016).

Las diferencias temporales, permite modelar el movimiento como una función de decaimiento a lo largo del tiempo, la cual se excita paulatinamente con los instantes de tiempo que un pixel presenta movimiento. Este método resulta útil debido a que la excitación en intervalos cortos que presenta movimiento permite mostrar un máximo en esta curva, la cual comienza a decaer en forma logarítmica. La rapidez en que decae la función está relacionado con la temporalidad del tiempo que el movimiento puede ser detectado. En términos prácticos la

diferencia temporal en el tiempo para un píxel en forma recursiva se define como:

$$T_{t-1}(\mathbf{x}) = \alpha T_t + (1 - \alpha) \times (I_t(\mathbf{x}) - I_{t-1}(\mathbf{x})) \tag{3.4}$$

donde $\mathbf{x}$ representa la posición del pixel, T es la función de decaimiento en el tiempo t, α es la constante de decaimiento, $(\mathbf{x}) - I_{t-1}(\mathbf{x}))$ es la contribución de la excitación de la presencia de movimiento en un par de imágenes (García-Huerta, Jiménez-Hernández, Herrera-Navarro, Hernández-Díaz, y Terol-Villalobos, 2014).

En la sustracción de fondo, el fondo se refiere a una región de la imagen que no incluye el primer plano (humano y/u objeto) y puede ser extraído completamente de la imagen, solo de alguna área del cuadro delimitador del humano, o una combinación de ambos (Ziaeefard y Bergevin, 2015). Esta técnica es ampliamente utilizada en aplicaciones de vigilancia y seguimiento debido a su eficacia para diferenciar rápidamente entre los elementos en movimiento y el fondo estático.

La sustracción de fondo puede complementarse con técnicas avanzadas de preprocesamiento y posprocesamiento, como el filtrado de ruido y el refinamiento de bordes, lo que mejora la precisión en la detección de movimiento y minimiza los falsos positivos. Este enfoque es especialmente valioso en escenarios donde mantener un modelo de fondo actualizado y preciso resulta fundamental para el análisis de escenas en tiempo real (Ziaeefard y Bergevin, 2015).

En la Figura. 3.8 diferentes imágenes extraídas de una secuencia de vídeo para ser analizadas en los distintos métodos para la detección de movimiento, donde en la Figura3.9 y Figura 3.10, se observan los métodos de diferencias temporales, sustracción de fondo y flujo óptico respectivamente.

La técnica de diferencia temporal se destaca por su capacidad para adaptarse bien a entornos dinámicos, siendo ideal para escenas donde los cambios son frecuentes y rápidos. Sin embargo, presenta limitaciones en la extracción precisa de todos los píxeles relevantes del objeto en movimiento. Por otro lado, la sustracción de fondo es eficaz en la extracción de información del objeto, aunque es más sensible a los cambios dinámicos del entorno (Valera y Velastín, 2005).

(a) (b) (c)

Figura 3.8: Muestra de fotogramas extraídos por una cámara.

(a) (b) (c)

Figura 3.9: Detección de movimiento con diferencias temporales.

(a) (b) (c)

Figura 3.10: Detección de movimiento con sustracción de fondo.

Además, la sustracción de fondo tiende a simplificar la clasificación al eliminar el contexto completo de la actividad o movimiento presente en la escena, lo que puede afectar su capacidad para distinguir acciones específicas en entornos ruidosos o con múltiples actividades simultáneas (Ziaeefard y Bergevin, 2015).

En conclusión, la técnica de diferencias temporales es adecuada para escenarios estáticos, mientras que la sustracción de fondo es más eficaz en aplicaciones de vigilancia y seguimiento. Ambas técnicas se complementan entre sí, y su elección depende de las condiciones específicas del entorno y los objetivos de la aplicación.

3.3 Metodología de implementación

En esta sección, se abordará la metodología de implementación del sistema propuesto, detallando los procesos para su uso. El enfoque se centra en la definición y aplicación de un lenguaje distribuido, el uso de primitivas de movimiento como base para la interpretación de actividades, y el uso del sistema distribuido multicámara.

La metodología se expone en la Figura 3.11, que ofrece una guía para la aplicación sistemática y precisa del para la utilización de SEL en un sistema distribuido, mostrado a continuación:

1. **Configuración y comunicación de componentes:** Como se observa en la Figura 3.12, la arquitectura definida para la implementación del sistema se compone de varias capas interconectadas, lo que garantiza una comunicación eficiente entre los distintos componentes. De acuerdo con esta estructura, el cliente inicia la comunicación desde la capa de presentación hacia la capa de control a través de una dirección o método de acceso.

   En la capa de control, se gestiona la información relacionada con los componentes o sensores disponibles, los cuales han sido previamente registrados en un directorio de direcciones. Este enfoque permite acceder a la información relevante de cada componente de manera rápida y eficiente, facilitando la transferencia de datos en tiempo real.

En la Figura 3.13 se ilustra el proceso de asignación de variables según los componentes disponibles en el sistema distribuido. En la capa de aplicación, el usuario interactúa con una interfaz que le permite visualizar los componentes registrados en el sistema distribuido. Durante esta fase inicial, se configuran las variables que referenciarán los componentes en el directorio de direcciones de la capa de control. Esta configuración optimiza la interacción del usuario con los recursos disponibles, facilitando la gestión y supervisión de los componentes en el sistema distribuido.

2. **Creación de programa:** Esta etapa comienza con el análisis del escenario, identificando las áreas y actividades relevantes. Como se muestra en la Figura 3.14, es fundamental conocer los escenarios, componentes y sensores para la creación de los programas.

   En primer lugar, el escenario se segmenta en estados, ajustando la complejidad según los requisitos de las actividades modeladas. A continuación, se evalúan las primitivas de movimiento más adecuadas, aplicando primitivas de secuencia para el seguimiento, concurrencia para la sincronización y paralelismo para detectar movimientos en áreas específicas.

   Finalmente, se desarrollan uno o más programas, segmentados según las primitivas de movimiento, los escenarios y las actividades, utilizando la gramática y sintaxis de SEL para modelar las actividades y facilitar su posterior inferencia.

3. **Comunicación inicial:** El cliente se comunica con la capa de control desde la capa de presentación mediante una dirección u otro método de acceso. La capa de control almacena información sobre los componentes o sensores disponibles, los cuales han sido previamente registrados en el directorio de direcciones. Esta información facilita la comunicación y transferencia de datos de manera efectiva y en cualquier momento.

4. **Solicitud y registro de componentes:** Cuando el cliente desea modelar e inferir actividades, se comunica con la capa de control para verificar la

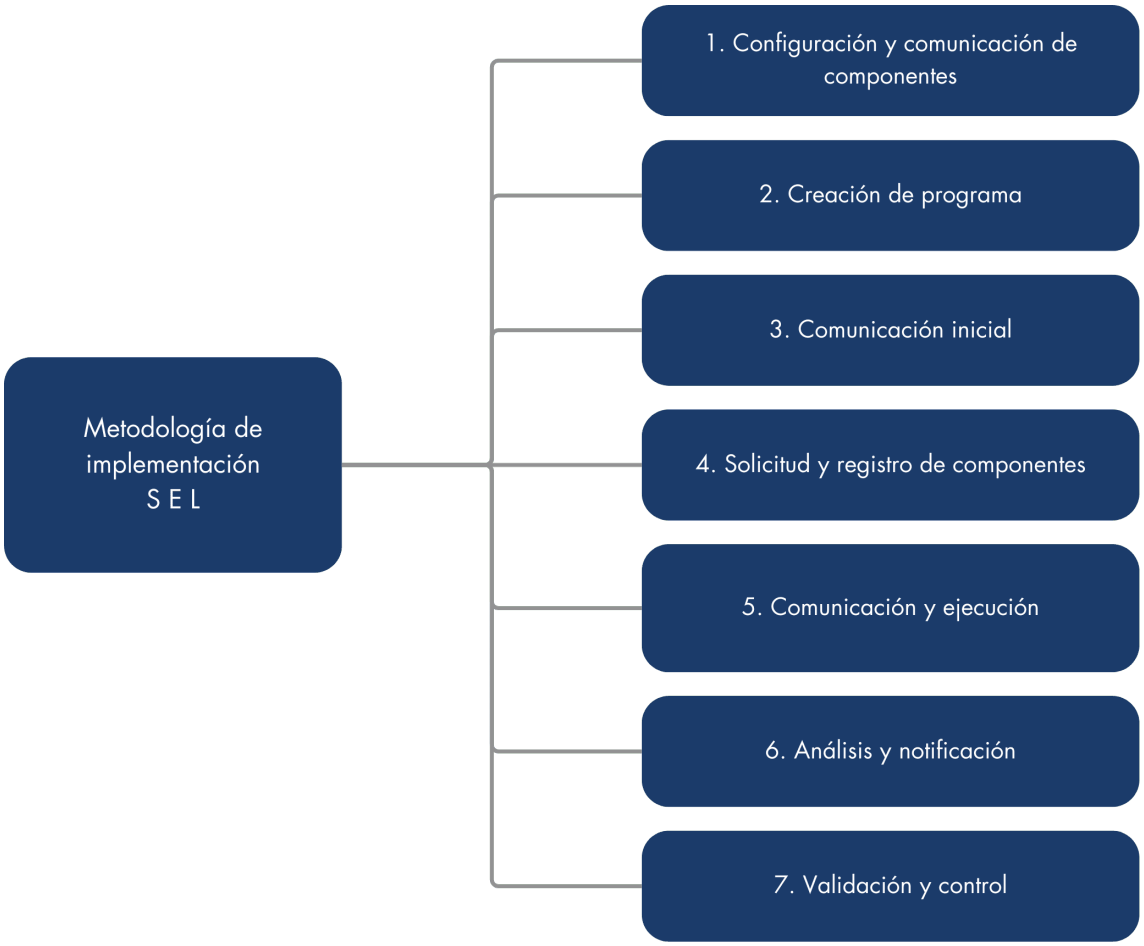


Figura 3.11: Metodología de implementación.

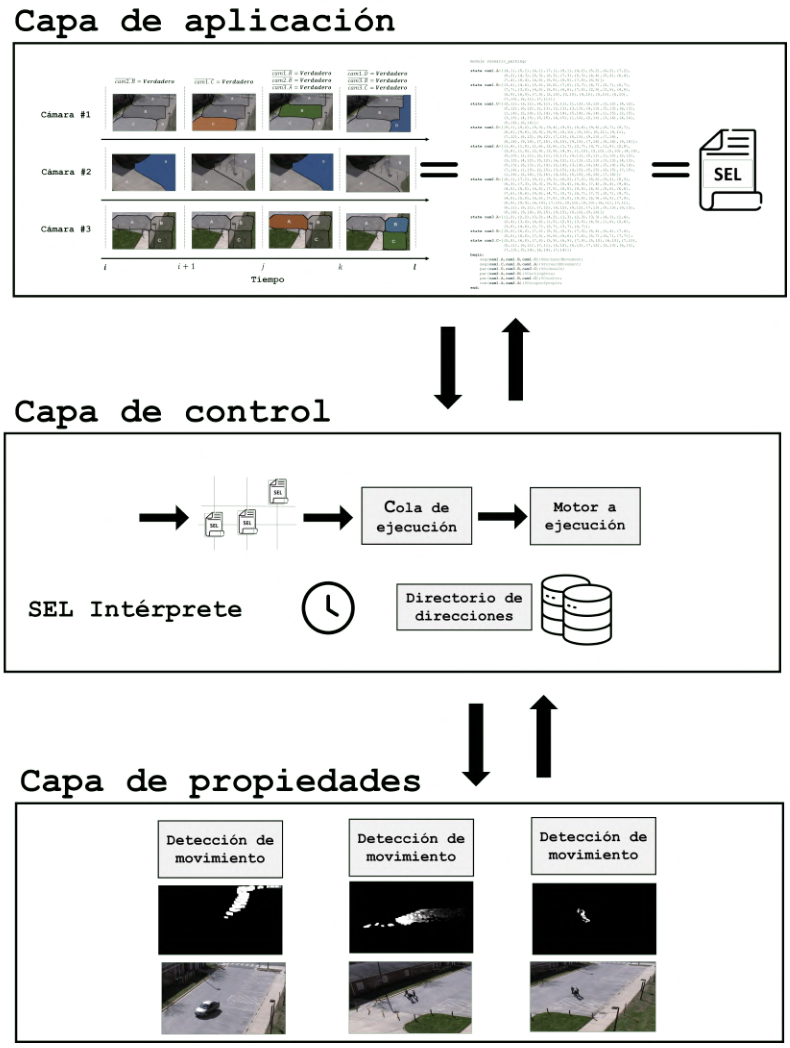


Figura 3.12: Arquitectura de sistema SEL.

disponibilidad de transferencia de información.

La capa central envía una lista con las características de los componentes y sensores, como la ubicación, el tipo de sensor y el método de detección de movimiento, sin revelar información sensible para el acceso. Con esta lista, el usuario puede escribir un programa de acuerdo con la semántica y las reglas gramaticales definidas en el lenguaje propuesto.

5. **Comunicación y ejecución:** La capa central traduce el programa y obtiene la información necesaria para la comunicación con cada componente del directorio de direcciones. Al interactuar con los componentes, recibe datos sobre las segmentaciones del espacio para detectar movimiento, de acuerdo con las características de cada sensor. Cada componente procesa la información o imagen obtenida y transfiere los datos del movimiento detectado a la capa de propiedades.

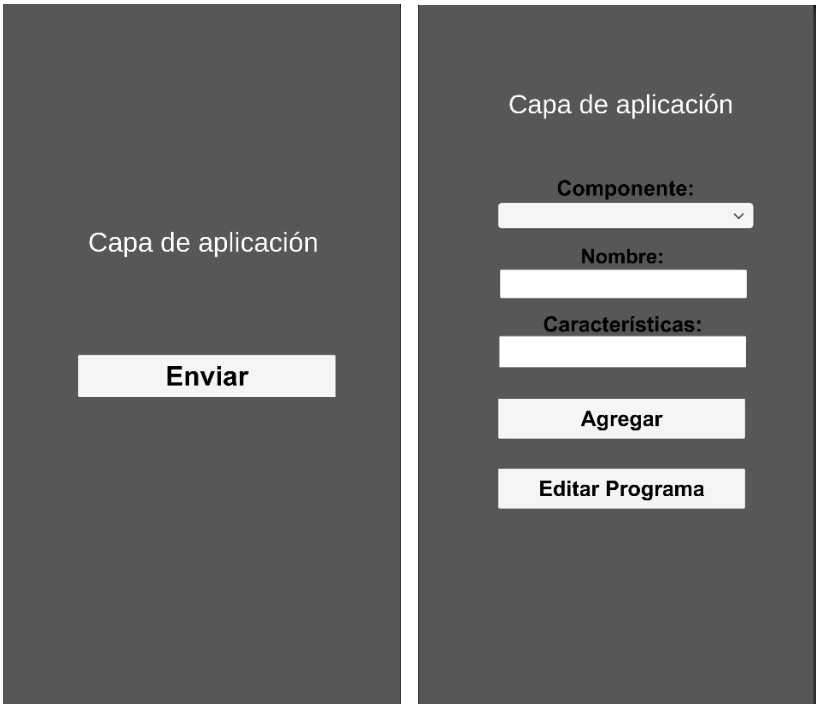
6. **Análisis y notificación:** Cuando los componentes detectan movimiento en las áreas designadas por el usuario en el programa, envían esta información a la capa central, indicando el momento y lugar del movimiento. La capa central analiza la información y, si coincide con las actividades descritas por el usuario, envía una notificación a la capa de presentación, informando que la actividad ha sido realizada.

7. **Validación y control:** Cada componente en el sistema distribuido, ubicado en la capa de propiedades, puede estar desarrollado en diferentes lenguajes o ser de distintos tipos de sensores. A pesar de las diferencias, todos se comunican entre sí según las solicitudes de servicio a través del sistema central de control, que gestiona todas las comunicaciones.

   Los componentes en la capa de propiedades capturan imágenes y las procesan para determinar el movimiento en áreas específicas, denominadas estados, utilizando métodos de detección de movimiento adecuados según las características del escenario. La información sobre el movimiento y el estado se envía al agente central.

La capa de aplicación valida la gramática SEL de los programas y envía la información validada a la capa de control, que posteriormente la transmite a cada componente de la capa de propiedades. Esta capa interactúa directamente con los usuarios, permitiéndoles consultar la información registrada.

Cuando la capa de control recibe y analiza la información sobre las áreas y momentos de movimiento, y si esta coincide con las actividades descritas, se envía una alerta correspondiente al usuario.



(a) (b)

(c) (d)

Figura 3.13: Capa de aplicación: Interfaz de usuario del sistema distribuido multicámara.

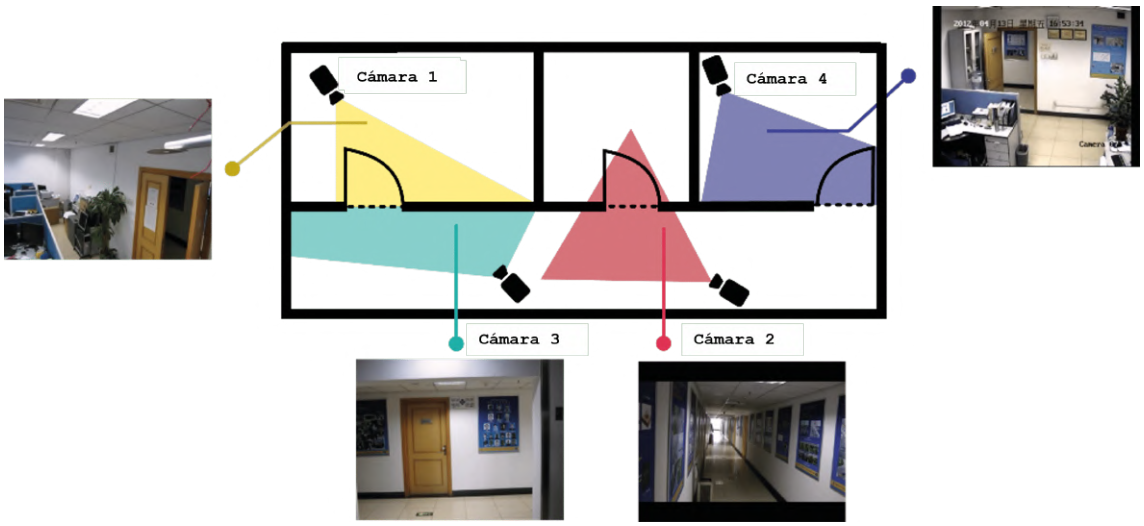


Figura 3.14: Distribución de las cámaras en el sistema distribuido.

Capítulo 4

Modelo experimental

En este capítulo se detallan las condiciones experimentales de acuerdo con la metodología propuesta, describiendo los elementos utilizados en diversas actividades y escenarios. Se presenta el uso del lenguaje en diferentes situaciones y perspectivas, donde el usuario o experto define y ejecuta los programas según la información disponible y los objetivos planteados.

Posteriormente, se realiza un análisis de los resultados obtenidos, aplicando los criterios establecidos en los programas. Estos resultados se comparan con los videos y las actividades detectadas por el experto. Finalmente, se presentan consideraciones finales sobre el sistema y su aplicación práctica.

Las primitivas de movimiento son fundamentales para determinar las actividades. Como se puede observar, las actividades definidas con la primitivas de secuencia muestran un mayor grado de precisión, ya que suelen realizarse con movimientos continuos. Sin embargo, las actividades basadas en concurrencia o paralelismo tienden a presentar un mayor margen de error debido a las características específicas de las actividades y el escenario, las cuales implican un menor movimiento.

4.1 Descripción experimental

En esta fase experimental, se emplearán todos los operadores primitivos de movimiento en cada entorno. Las actividades modeladas varían en complejidad debido a los componentes involucrados y al número de escenarios. Cada actividad se identifica con un código único, lo que facilita su seguimiento temporal.

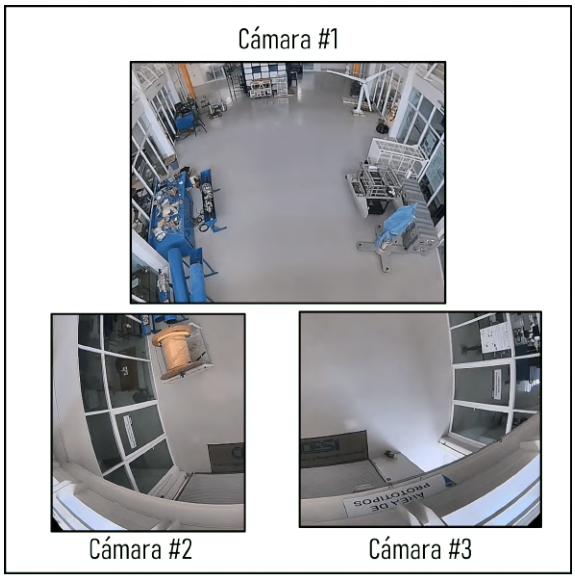
Los tres escenarios se clasifican en: escenario continuo de personas, escenario continuo de vehículos y escenario complejo, como se muestra en la Figura 4.1.

- **Escenario continuo de personas:** Este escenario se caracteriza por abarcar un área no solapada, cubierta por tres cámaras.Los escenarios presentan desafíos como los cambios en la iluminación y los reflejos, que pueden influir en la efectividad de los métodos de detección de movimiento. Sin embargo, la segmentación que se aplica es fundamental para reducir estos efectos. Los movimientos captados en áreas específicas de las cámaras permiten contabilizar con precisión los objetos, personas o grupos de personas presentes en el escenario.

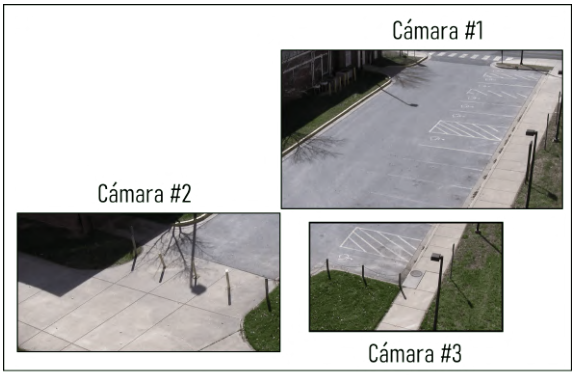
- **Escenario continuo de vehículos:** Similar al escenario anterior, este también cubre un área no solapada mediante las cámaras. Sin embargo, la situación cambia según el contexto y la perspectiva, ya que se desarrolla en un entorno al aire libre. Un aspecto importante en este escenario son las variaciones en la luz, que afectan la detección de movimiento.

- **Escenario complejo:** Este escenario se considera complejo debido a la existencia de áreas discontinuas, lo que impide obtener información en algunas zonas. Es importante tener esto en cuenta al modelar las actividades, especialmente durante el análisis de trayectorias. Además, hay un movimiento de fondo constante, similar al de un escenario con personas, pero esto no afecta al sistema gracias a la segmentación que se ha implementado en cada caso.

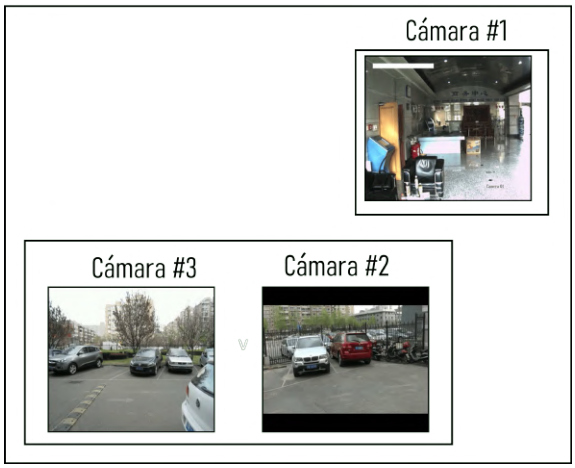
La parte experimental incluye tres entornos distintos, cada uno con su propio conjunto de cámaras, diseñados con características y objetivos específicos para las actividades a modelar. En esta fase, se emplearán todos los operadores primitivos de movimiento en cada entorno. Las actividades modeladas variarán en complejidad debido a los diferentes componentes involucrados y al número de escenarios. Cada actividad modelada se identificará con un código único para su



(a) Escenario continuo de personas.



(b) Escenario continuo de vehículos.

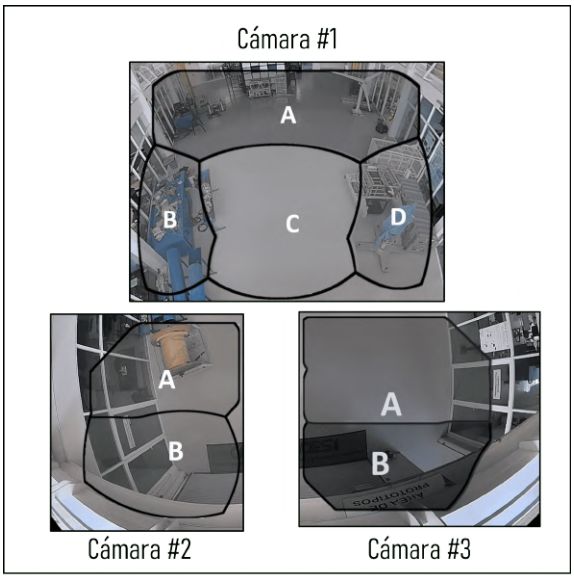


(c) Escenario complejo.

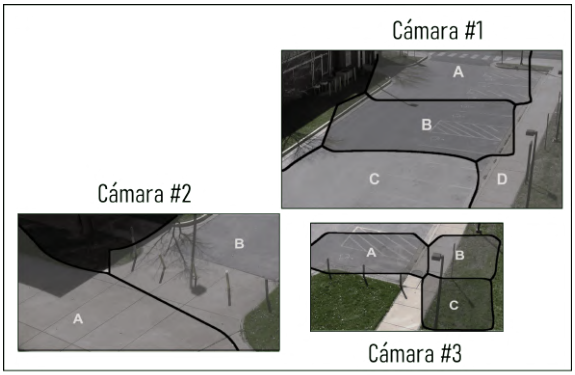
Figura 4.1: Escenarios de la parte experimental.

seguimiento temporal.

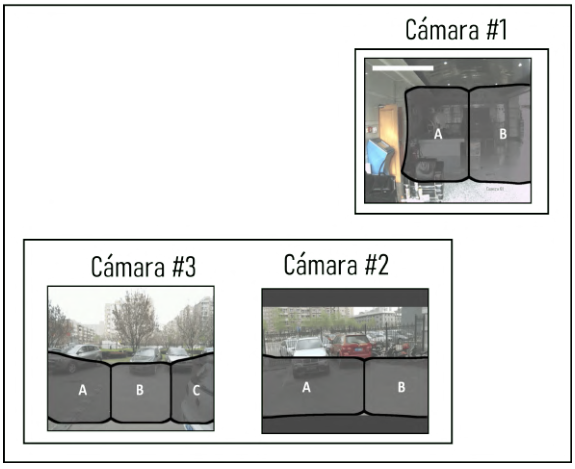
Todos los componentes de los escenarios propuestos se segmentarán según el modelado de las actividades que se desean inferir, como se muestra en la Figura 4.2. Las sentencias del modelado de actividades con el lenguaje SEL, de acuerdo con la segmentación dada en cada entorno, así como los nombres de referencia de las actividades, se pueden ver en la Figura 4.3, y por último, en la Figura 4.4, Figura 4.5 y Figura 4.6 muestran los códigos completos que se utilizaron para modelar actividades.



(a) Escenario continuo de personas.



(b) Escenario continuo de vehículos.



(c) Escenario complejo.

Figura 4.2: Escenarios segmentados en la parte experimental.

```
par(cam1.A,cam1.B,cam2.A,cam3.A);%Main Area;
par(cam1.B,cam1.D);%Workplace;
seq(cam3.B,cam3.A,cam1.C,cam1.A);%Sequence1;
seq(cam1.A,cam1.C,cam3.A,cam3.B);%Sequence2;
con(cam2.B,cam3.B);%Door Movement;
```

(a) Escenario continuo de personas.

```
seq(cam3.B,cam3.C,null,cam2.A,cam2.B,null,cam1.A,cam1.B);%CameraSequence;
par(cam3.A,cam3.B,cam3.C,cam2.A,cam2.B);%ParkingMotion;
con(cam1.A,cam1.B);%CafeteriaArea;
```

(b) Escenario continuo de vehículos.

```
seq(cam1.A,cam1.B,cam1.C);%BackwardMovement;
seq(cam1.C,cam1.B,cam1.A);%ForwardMovement;
par(cam1.D,cam3.B,cam3.C);%Sidewalk;
par(cam2.A,cam2.B);%ParkingArea;
par(cam1.A,cam1.B,cam1.C);%Counter;
con(cam1.A,cam2.A);%Groupsofpeople;
```

(c) Escenario complejo.

Figura 4.3: Código de actividades de los escenarios.

```
module scenario_workArea;

state cam1.A=[(2,2),(3,2),(4,2),(5,2),(6,2),(7,2),(8,2),(2,3),(3,3),
              (4,3),(5,3),(6,3),(7,3),(8,3),(2,4),(3,4),(4,4),(5,4),
              (6,4),(7,4),(8,4),(2,5),(3,5),(4,5),(5,5),(6,5),(7,5),
              (8,5),(2,6),(3,6),(6,6),(7,6),(8,6)];

state cam1.B=[(1,7),(2,7),(1,8),(2,8),(1,9),(2,9),(1,10),(2,10),(1,11),
              (2,11),(1,12),(2,12),(1,13),(2,13),(1,14),(2,14),
              (1,15),(2,15)];

state cam1.C=[(4,6),(5,6),(3,7),(4,7),(5,7),(6,7),(7,7),(3,8),(4,8),(5,8),
              (6,8),(7,8),(3,9),(4,9),(5,9),(6,9),(7,9),(3,10),(4,10),
              (5,10),(6,10),(7,10),(3,11),(4,11),(5,11),(6,11),(3,12),
              (4,12),(5,12),(6,12),(3,13),(4,13),(5,13),(6,13),(3,14),
              (4,14),(5,14),(6,14),(3,15),(4,15),(5,15),(6,15),(5,16)];

state cam1.D=[(8,7),(9,7),(8,8),(9,8),(8,9),(9,9),(8,10),(9,10),
              (7,11),(8,11),(9,11),(7,12),(8,12),(9,12),(7,13),(8,13),
              (9,13),(7,14),(8,14),(9,14),(8,15)];

state cam2.A=[(5,2),(6,2),(7,2),(8,2),(9,2),(4,3),(5,3),(6,3),(7,3),
              (8,3),(9,3),(4,4),(5,4),(6,4),(7,4),(8,4),(9,4),(3,5),
              (4,5),(5,5),(6,5),(7,5),(8,5),(9,5),(3,6),(4,6),(5,6),
              (6,6),(7,6),(8,6),(9,6),(3,7),(4,7),(5,7),(6,7),(7,7),
              (8,7),(9,7)];

state cam2.B=[(3,8),(4,8),(5,8),(6,8),(7,8),(8,8),(9,8),(3,9),(4,9),
              (5,9),(6,9),(7,9),(8,9),(9,9),(3,10),(4,10),(5,10),
              (6,10),(7,10),(8,10),(9,10),(3,11),(4,11),(5,11),
              (6,11),(7,11),(8,11),(9,11),(3,12),(4,12),(5,12),
              (6,12),(7,12),(8,12),(9,12),(3,13),(4,13),(5,13),
              (6,13),(7,13),(8,13),(9,13),(4,14),(5,14),(6,14),
              (7,14),(8,14)];

state cam3.A=[(1,1),(2,1),(3,1),(4,1),(5,1),(6,1),(7,1),(1,2),
              (2,2),(3,2),(4,2),(5,2),(6,2),(7,2),(1,3),(2,3),
              (3,3),(4,3),(5,3),(6,3),(7,3),(1,4),(2,4),(3,4),
              (4,4),(5,4),(6,4),(7,4),(1,5),(2,5),(3,5),(4,5),
              (5,5),(6,5),(7,5),(1,6),(2,6),(3,6),(4,6),(5,6),
              (6,6),(7,6),(1,7),(2,7),(3,7),(4,7),(5,7),(6,7),
              (7,7),(2,8),(3,8),(4,8),(5,8)];

state cam3.B=[(1,8),(6,8),(7,8),(1,9),(2,9),(3,9),(4,9),(5,9),
              (6,9),(7,9),(1,10),(2,10),(3,10),(4,10),(5,10),
              (6,10),(7,10),(1,11),(2,11),(3,11),(4,11),(5,11),
              (6,11),(7,11),(1,12),(2,12),(3,12),(4,12),(5,12),
              (6,12),(2,13),(3,13),(4,13),(5,13),(6,13)];


begin;
      par(cam1.A,cam1.B,cam2.A,cam3.A);%Main Area;
      par(cam1.B,cam1.D);%Workplace;
      seq(cam3.B,cam3.A,cam1.C,cam1.A);%Sequence1;
      seq(cam1.A,cam1.C,cam3.A,cam3.B);%Sequence2;
      con(cam2.B,cam3.B);%Door Movement;
end;
```

Figura 4.4: Código de escenario continuo de personas.

```
module scenario_parking;

state cam1.A=[(4,1),(5,1),(6,1),(7,1),(8,1),(4,2),(5,2),(6,2),(7,2),
              (8,2),(4,3),(5,3),(6,3),(7,3),(8,3),(4,4),(5,4),(6,4),
              (7,4),(8,4),(4,5),(5,5),(6,5),(7,5),(8,5)];
state cam1.B=[(3,6),(4,6),(5,6),(6,6),(7,6),(3,7),(4,7),(5,7),(6,7),
              (7,7),(3,8),(4,8),(5,8),(6,8),(7,8),(2,9),(3,9),(4,9),
              (5,9),(6,9),(7,9),(2,10),(3,10),(4,10),(5,10),(6,10),
              (7,10),(6,11),(7,11)];
state cam1.C=[(2,11),(3,11),(4,11),(5,11),(1,12),(2,12),(3,12),(4,12),
              (5,12),(6,12),(1,13),(2,13),(3,13),(4,13),(5,13),(6,13),
              (1,14),(2,14),(3,14),(4,14),(5,14),(6,14),(1,15),(2,15),
              (3,15),(4,15),(5,15),(6,15),(1,16),(2,16),(3,16),(4,16),
              (5,16),(6,16)];
state cam1.D=[(9,1),(9,2),(9,3),(9,4),(9,5),(8,6),(9,6),(8,7),(9,7),
              (8,8),(9,8),(8,9),(9,9),(8,10),(9,10),(8,11),(9,11),
              (7,12),(8,12),(9,12),(7,13),(8,13),(9,13),(7,14),
              (8,14),(9,14),(7,15),(8,15),(9,15),(7,16),(8,16),(9,16)];
state cam2.A=[(1,4),(1,5),(1,6),(2,6),(1,7),(2,7),(3,7),(1,8),(2,8),
              (3,8),(1,9),(2,9),(3,9),(4,9),(1,10),(2,10),(3,10),(4,10),
              (5,10),(1,11),(2,11),(3,11),(4,11),(5,11),(1,12),(2,12),
              (3,12),(4,12),(5,12),(6,12),(1,13),(2,13),(3,13),(4,13),
              (5,13),(6,13),(1,14),(2,14),(3,14),(4,14),(5,14),(6,14),
              (7,14),(1,15),(2,15),(3,15),(4,15),(5,15),(6,15),(7,15),
              (1,16),(2,16),(3,16),(4,16),(5,16),(6,16),(7,16)];
state cam2.B=[(6,1),(7,1),(8,1),(9,1),(6,2),(7,2),(8,2),(9,2),(5,3),
              (6,3),(7,3),(8,3),(9,3),(5,4),(6,4),(7,4),(8,4),(9,4),
              (4,5),(5,5),(6,5),(7,5),(8,5),(9,5),(4,6),(5,6),(6,6),
              (7,6),(8,6),(9,6),(4,7),(5,7),(6,7),(7,7),(8,7),(9,7),
              (4,8),(5,8),(6,8),(7,8),(8,8),(9,8),(5,9),(6,9),(7,9),
              (8,9),(9,9),(6,10),(7,10),(8,10),(9,10),(6,11),(7,11),
              (8,11),(9,11),(7,12),(8,12),(9,12),(7,13),(8,13),(9,13),
              (8,14),(9,14),(8,15),(9,15),(8,16),(9,16)];
state cam3.A=[(1,2),(2,2),(3,2),(4,2),(1,3),(2,3),(3,3),(4,3),(1,4),
              (2,4),(3,4),(4,4),(1,5),(2,5),(3,5),(4,5),(1,6),(2,6),
              (3,6),(4,6),(1,7),(2,7),(3,7),(4,7)];
state cam3.B=[(5,2),(6,2),(7,2),(5,3),(6,3),(7,3),(5,4),(6,4),(7,4),
              (5,5),(6,5),(7,5),(5,6),(6,6),(7,6),(5,7),(6,7),(7,7)];
state cam3.C=[(5,8),(6,8),(7,8),(5,9),(6,9),(7,9),(5,10),(6,10),(7,10),
              (5,11),(6,11),(7,11),(5,12),(6,12),(7,12),(5,13),(6,13),
              (7,13),(5,14),(6,14),(7,14)];
begin;
    seq(cam1.A,cam1.B,cam1.C);%BackwardMovement;
    seq(cam1.C,cam1.B,cam1.A);%ForwardMovement;
    par(cam1.D,cam3.B,cam3.C);%Sidewalk;
    par(cam2.A,cam2.B);%ParkingArea;
    par(cam1.A,cam1.B,cam1.C);%Counter;
    con(cam1.A,cam2.A);%Groupsofpeople;
end;
```

Figura 4.5: Código de escenario continuo de vehículos.

```
module scenario_complex;

state cam1.A=[(3,4),(4,4),(5,4),(6,4),(3,5),(4,5),(5,5),(6,5),(3,6),(4,6),
              (5,6),(6,6),(3,7),(4,7),(5,7),(6,7),(3,8),(4,8),(5,8),(6,8),
              (3,9),(4,9),(5,9),(6,9),(3,10),(4,10),(5,10),(6,10),(3,11),
              (4,11),(5,11),(6,11),(3,12),(4,12),(5,12),(6,12),(3,13),
              (4,13),(5,13),(6,13),(3,14),(4,14),(5,14),(6,14)];
state cam1.B=[(7,4),(8,4),(9,4),(7,5),(8,5),(9,5),(7,6),(8,6),(9,6),(7,7),(8,7),
              (9,7),(7,8),(8,8),(9,8),(7,9),(8,9),(9,9),(7,10),(8,10),(9,10),
              (7,11),(8,11),(9,11),(7,12),(8,12),(9,12),(7,13),(8,13),(9,13),
              (7,14),(8,14),(9,14)];
state cam2.A=[(1,9),(2,9),(3,9),(4,9),(5,9),(1,10),(2,10),(3,10),(4,10),(5,10),
              (1,11),(2,11),(3,11),(4,11),(5,11),(1,12),(2,12),(3,12),(4,12),
              (5,12),(1,13),(2,13),(3,13),(4,13),(5,13),(1,14),(2,14),(3,14),
              (4,14),(5,14)];
state cam2.B=[(6,9),(7,9),(8,9),(9,9),(6,10),(7,10),(8,10),(9,10),(6,11),(7,11),
              (8,11),(9,11),(6,12),(7,12),(8,12),(9,12),(6,13),(7,13),(8,13),
              (9,13),(6,14),(7,14),(8,14),(9,14)];
state cam3.A=[(1,9),(2,9),(3,9),(1,10),(2,10),(3,10),(1,11),(2,11),(3,11),(1,12),
              (2,12),(3,12),(1,13),(2,13),(3,13),(1,14),(2,14),(3,14),(1,15),(2,15),(3,
state cam3.B=[(4,9),(5,9),(6,9),(7,9),(4,10),(5,10),(6,10),(7,10),(4,11),(5,11),(6,11),
              (7,11),(4,12),(5,12),(6,12),(7,12),(4,13),(5,13),(6,13),(7,13),(4,14),
              (5,14),(6,14),(7,14),(4,15),(5,15),(6,15),(7,15)];
state cam3.C=[(8,9),(9,9),(8,10),(9,10),(8,11),(9,11),(8,12),(9,12),(8,13),
              (9,13),(8,14),(9,14),(8,15),(9,15)];

begin;
     seq(cam3.B,cam3.C,null,cam2.A,cam2.B,null,cam1.A,cam1.B);%CameraSequence;
     par(cam3.A,cam3.B,cam3.C,cam2.A,cam2.B);%ParkingMotion;
     con(cam1.A,cam1.B);%CafeteriaArea;
end;
```

Figura 4.6: Código de escenario complejo.

Capítulo 5

Resultados

En este capítulo se presentan y analizan los resultados obtenidos en la experimentación en los sistemas distribuido multicámara con el lenguaje propuesto en los tres entornos evaluados, cada uno con su conjunto específico de cámaras y características particulares. Se evaluó la efectividad de las primitivas de movimiento aplicados en cada entorno para modelar actividades con diferentes niveles de complejidad.

Los resultados muestran que el sistema propuesto es capaz de identificar y detectar las actividades modeladas, utilizando la segmentación adecuada y el nombramiento correcto de variables para los componentes disponibles. Además, se discuten las ventajas y limitaciones, así como las posibles mejoras y aplicaciones futuras del sistema. Los datos recopilados ofrecen una base sólida para futuras investigaciones y desarrollos en el ámbito de la detección y el modelado de actividades en entornos distribuidos.

En la fase experimental, se analizaron diversas actividades en los tres entornos previamente descritos. Durante esta etapa, se emplearon todos los operadores primitivos de movimiento en cada entorno, lo que permitió modelar actividades con diferentes niveles de complejidad. Estas complejidades se derivaron no solo de los componentes involucrados, sino también del número de escenarios considerados y sus características específicas. Cada actividad modelada fue asignada con un código único, lo que facilitó su seguimiento detallado y su análisis temporal a lo largo del proceso experimental.

En la sección de resultados, se presentan tablas correspondientes a cada uno de los escenarios descritos anteriormente (ver Tabla 1, 2 y 3).

En las Tabla 1, 2 y 3, se describe la cantidad de fotogramas analizados durante la búsqueda de actividades, así como la utilización de dos métodos diferentes de detección de movimiento. Cada entorno propuesto en la fase experimental incluye el modelado de diversas actividades, a las cuales se les asignó un nombre, y cada una utiliza una primitiva de movimiento, como se ilustra en la Figura 4.3.

Se trabajó considerando el 100 % de las actividades detectables en el sistema, evaluando cuántas fueron correctamente detectadas (verdadero positivo, VP), cuántas no fueron detectadas (falso negativo, FN), cuántas fueron correctamente descartadas como no presentes (verdadero negativo, VN), y cuántas fueron incorrectamente identificadas como existentes (falso positivo, FP). Estos datos permiten analizar el desempeño del sistema en términos de métricas como la precisión, la sensibilidad y la tasa de error.

Estas actividades fueron registradas de manera efectiva gracias al sistema distribuido multicámara, el cual permitió la recolección, procesamiento y análisis de los datos. Este sistema facilitó no solo la detección de movimiento, sino también la inferencia de las actividades modeladas en cada escenario, lo que permitió una evaluación detallada de su ejecución en condiciones operativas reales.

Durante la fase experimental, se probaron dos técnicas de detección de movimiento: diferencias temporales y sustracción de fondo. Estas técnicas, cuyos resultados pueden variar debido a diversos factores externos, fueron analizadas en las tablas presentadas. Es importante considerar ambas metodologías, ya que ofrecen ventajas complementarias y se adaptan a las características específicas de distintos escenarios. Sin embargo, algunos entornos presentan variables no controlables, como cambios en la iluminación, sombras, reflejos o movimientos inesperados, que pueden influir en su desempeño.

Cada técnica aporta beneficios únicos al enfrentar condiciones variables, optimizando la detección y el seguimiento de actividades en sistemas de vigilancia. Para evaluar el desempeño de cada técnica de detección de movimiento, se analizaron las actividades detectadas con el objetivo de identificar posibles

falsos positivos (FP) y falsos negativos (FN). Con base en estos resultados, se calcularon métricas clave como la precisión y la exactitud, las cuales ofrecen una valoración integral de la efectividad y confiabilidad de cada técnica.

La precisión mide la proporción de verdaderos positivos entre todas las predicciones positivas realizadas por el modelo, lo que refleja la exactitud de estas predicciones. En cambio, el *recall* (sensibilidad) representa la proporción de verdaderos positivos identificados correctamente entre todos los casos positivos reales, lo que indica la capacidad del modelo para detectar todos los casos positivos posibles

Estas métricas ofrecen una evaluación detallada y comparativa del rendimiento de cada técnica, lo que permite identificar cuál es la más efectiva bajo diferentes escenarios y condiciones experimentales

Los resultados obtenidos en este estudio se presentan de manera gráfica mediante las curvas ROC (Receiver Operating Characteristic). Estas curvas son herramientas esenciales en la evaluación de modelos de clasificación, ya que ilustran la relación entre la tasa de verdaderos positivos (*TPR*, por sus siglas en inglés) y la tasa de falsos positivos (*FPR*) para diferentes umbrales de decisión. Las curvas ROC permiten evaluar la capacidad discriminativa del modelo, siendo que una curva que se aproxima al vértice superior izquierdo del gráfico indica un mejor rendimiento, lo que refleja una mayor capacidad para clasificar correctamente las clases positivas y negativas.

En la Figura 5.1, se presentan de manera clara y accesible las comparativas entre los métodos utilizados para cada actividad. El entorno que muestra los resultados menos favorables es el escenario continuo de vehículos, debido a factores externos como sombras y cambios en la luminosidad. Además, al analizar los vídeos, se observa que muchos de los movimientos no son constantes, lo que afecta los resultados, especialmente con la técnica de diferencias temporales. En los otros entornos, las diferencias entre las técnicas son mínimas y los resultados obtenidos son satisfactorios. En cuanto a *precisión* y *exactitud*, los resultados obtenidos al modelar actividades multicámara.

Tabla 5.1: Resultados del escenario continuo de personas.

TM[1]	Fotograma	Actividad	SEL (TP[4]) %	GT[5] %	FP[6] %	FN[7] %	VN[8] %	Precision %	Recall %
SF[2]	107943	Main Area	25.2	28.0	6.0	2.8	66.0	80.7	90.1
		Workplace	27.7	21.8	3.6	5.9	62.8	88.6	82.4
		Sequence1	10.8	13.9	1.8	3.1	84.3	85.7	77.9
		Sequence2	6.2	9.2	1.4	3.1	89.3	81.0	66.7
		Door Movement	24.3	18.0	1.4	6.3	67.9	94.5	79.4
DT[3]	107943	Main Area	20.0	28.7	3.7	8.7	67.6	84.3	69.8
		Workplace	17.5	24.8	3.7	7.3	71.5	82.5	70.7
		Sequence1	11.1	14.0	1.1	2.9	84.9	91.0	79.2
		Sequence2	7.3	9.3	1.3	2.0	89.5	85.1	78.4
		Door Movement	22.4	19.4	1.4	3.1	73.1	94.3	88.0

[1] Técnica de detección de movimiento.
[2] Sustracción de fondo.
[3] Diferencias temporales.
[4] Verdadero positivo.
[5] Datos de referencia (Ground Truth).
[6] Falso positivo.
[7] Falso negativo.
[8] Verdadero negativo.

La segmentación de los estados puede resultar una tarea compleja para el usuario, ya que requiere atención cuidadosa para asegurar que las actividades definidas se identifiquen correctamente, evitando la detección errónea de actividades. En este sentido, se recomienda considerar el uso de métodos de segmentación automática que faciliten la definición de los estados.

Los principales factores que afectan al sistema incluyen la disminución del movimiento en los objetos que realizan las actividades. La pérdida de movimiento puede generar la identificación de múltiples actividades o la falta de una actividad completa. Además, factores como la variación en la luminosidad, los reflejos y la intersección de objetos o de los estados definidos en las actividades también influyen en el rendimiento del sistema.

Las primitivas de movimiento son herramientas útiles para el modelado e inferencia de actividades. Como se ha observado, las actividades modeladas con primitivas secuenciales presentan un mayor grado de precisión, ya que generalmente implican movimientos continuos. En cambio, las actividades basadas en concurrencia o paralelismo suelen mostrar mayores márgenes de error debido a las características propias de las actividades, que tienden a tener movimientos más limitados.

Tabla 5.2: Resultado de escenario continuo de vehículos.

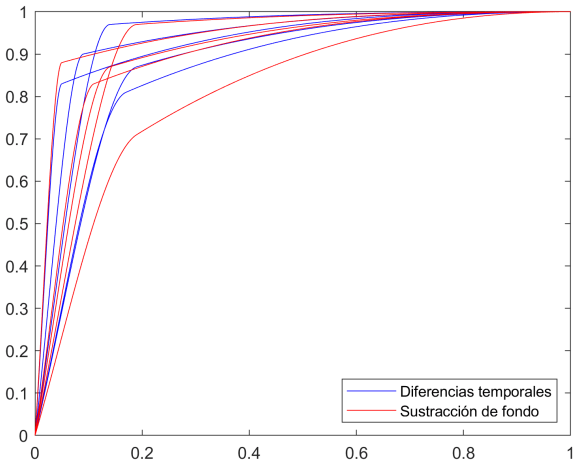
TM[1]	Fotograma	Actividad	SEL (TP[4]) %	GT[5] %	FP[6] %	FN[7] %	VN[8] %	Precision %	Recall %
SF[2]	126266	Backward movement	14.8	20.2	5.8	5.4	74.0	71.7	73.3
		Forward movement	17.7	17.9	3.4	0.2	78.6	83.7	98.7
		Sidewalk	10.6	5.0	1.5	5.6	82.3	87.9	65.4
		Parking Area	23.0	13.9	0.6	9.1	67.3	97.5	71.7
		Counter	26.5	25.9	3.4	0.7	69.4	88.6	97.5
		Groups of people	7.9	9.8	2.1	1.9	88.1	79.1	81.0
DT[3]	126266	Backward movement	22.7	19.3	3.2	3.4	70.7	87.6	86.9
		Forward movement	18.7	17.2	4.4	1.5	75.3	81.0	92.4
		Sidewalk	11.1	4.9	1.2	6.2	81.5	90.0	64.3
		Parking Area	24.6	13.0	0.7	11.6	63.0	97.1	68.0
		Counter	24.6	13.0	0.7	11.6	63.0	97.1	68.0
		Groups of people	15.0	8.4	3.0	6.6	75.4	83.3	69.4

[1] Técnica de detección de movimiento.
[2] Sustracción de fondo.
[3] Diferencias temporales.
[4] Verdadero positivo.
[5] Datos de referencia (Ground Truth).
[6] Falso positivo.
[7] Falso negativo.
[8] Verdadero negativo.

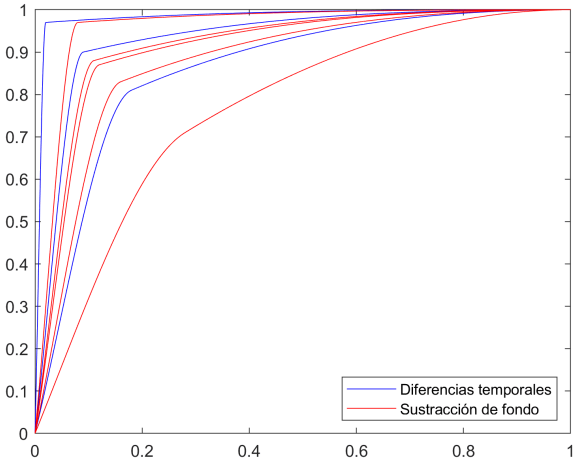
Tabla 5.3: Resultados de escenario complejo

TM[1]	Fotograma	Actividad	SEL (TP[4]) %	GT[5] %	FP[6] %	FN[7] %	VN[8] %	Precision %	Recall %
SF[2]	3600	Camera Sequence	24.8	29.3	6.5	4.5	64.3	79.4	84.7
		Parking Motion	33.1	22.7	8.4	10.5	47.9	79.7	76.0
		Cafeteria Area	25.8	34.6	5.5	8.8	59.9	82.4	74.6
DT[3]	3600	Camera Sequence	23.8	22.5	4.8	1.3	70.1	83.3	94.8
		Parking Motion	40.4	30.2	9.4	10.2	40.1	81.2	79.9
		Cafeteria Area	18.9	30.4	7.2	11.5	62.5	72.5	62.1

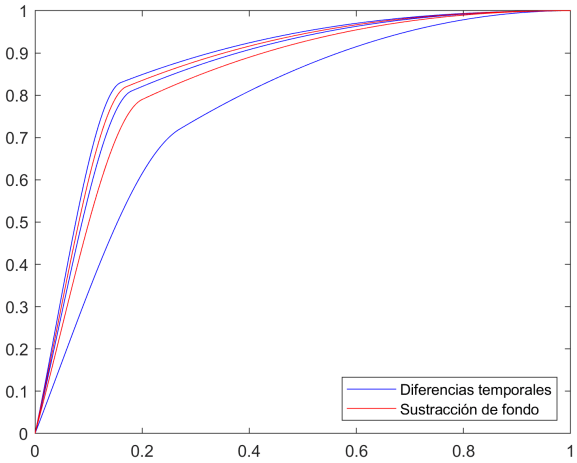
[1] Técnica de detección de movimiento.
[2] Sustracción de fondo.
[3] Diferencias temporales.
[4] Verdadero positivo.
[5] Datos de referencia (Ground Truth).
[6] Falso positivo.
[7] Falso negativo.
[8] Verdadero negativo.



(a) Escenario continuo de personas.



(b) Escenario continuo de vehículos.



(c) Escenario complejo.

Figura 5.1: Gráfica ROC de escenarios.

El lenguaje y sus componentes proporcionan flexibilidad para su implementación en una amplia variedad de escenarios, siempre y cuando se tenga conocimiento de los factores no controlables que pueden afectar la detección de actividades. Esto permite una configuración más eficiente del sistema, adaptándose con facilidad a diferentes condiciones del entorno.

En conclusión, los resultados obtenidos demuestran la efectividad del modelo en la identificación tanto de instancias positivas como negativas. La tasa de falsos positivos se mantuvo por debajo del 10 %, lo que indica una baja cantidad de errores al clasificar instancias negativas como positivas. Por otro lado, los falsos negativos alcanzaron un máximo del 11 %, lo que refleja una precisión moderada en la clasificación de instancias positivas. Los verdaderos negativos fluctuaron entre el 60 % y el 80 %, lo que resalta la capacidad del modelo para identificar correctamente las instancias negativas. La precisión general del modelo varió entre el 80 % y el 90 %, lo que muestra una eficacia destacada. Finalmente, el recall(sensibilidad) alcanzó un rango de 62 % a 97 %, subrayando la alta efectividad del modelo para detectar instancias positivas. Estos resultados sugieren que, a pesar de ciertas limitaciones, el modelo presenta un rendimiento robusto y confiable en la mayoría de las pruebas realizadas.

Capítulo 6

Conclusiones

Este capítulo describe las conclusiones obtenidas de la investigación llevada a cabo sobre el modelado de actividades multicámara mediante un lenguaje distribuido SEL. Durante el estudio, se examinaron diversos escenarios y métodos de detección de movimiento para valorar la efectividad y exactitud del sistema. Los resultados proporcionan una perspectiva precisa de las fortalezas y áreas de mejora en la implementación de este método para la supervisión y el estudio de actividades en diversos contextos. A continuación, se describen las conclusiones más significativas y las consecuencias de los descubrimientos para futuros estudios y usos prácticos.

El uso de un lenguaje para el manejo de un sistema de vigilancia resulta ser de gran ayuda, especialmente para aquellos usuarios finales sin habilidades técnicas, ya que facilita la tarea de modelar y obtener resultados de manera sencilla. Esto se logra mediante el conocimiento de la semántica y gramática del lenguaje, lo que permite manejarlo de acuerdo con las necesidades o la información que se busca o requiere.

El trabajo realizado ha demostrado la eficacia del lenguaje de descripción de eventos en sistemas distribuido multicámara para determinar y modelar actividades en un sistema multicámara distribuido. SEL permite la activación de los estados de movimiento en el espacio y el tiempo en uno o varios contextos, lo que permite una extensa variedad de usos en la visión computacional.

La implementación de operadores de movimiento primitivos ha jugado un papel crucial en la descripción del movimiento, facilitando la identificación de movimiento en estados determinados y sincronizando estos estados a lo largo de uno o más escenario.

Estos operadores también establecen un límite temporal en la progresión del movimiento, ofreciendo una descripción precisa y detallada.

El uso de un sistema de vigilancia distribuido con el lenguaje SEL puede ser de gran utilidad para usuarios finales no expertos, ya que simplifica la tarea de modelar e inferir actividades o comportamientos en múltiples escenarios. La arquitectura en capas del sistema permite una comunicación clara y eficiente entre cada capa, asegurando que la información sea accesible para el usuario. Esto también proporciona un mayor rendimiento, modularidad, tolerancia a fallos y escalabilidad.

La estructura y modularidad del sistema de vigilancia distribuido son sus principales fortalezas. La implementación del sistema ofrece un mayor alcance en términos de usuarios y componentes analizados, y el uso de SEL facilita el modelado e inferencia de actividades complejas sin necesidad de métodos estocásticos o probabilísticos especializados.

En la parte experimental, se probaron diversos escenarios y técnicas de detección de movimiento, validando la confiabilidad del lenguaje SEL en diferentes contextos y configuraciones de actividades. Las pruebas mostraron que, aunque algunos entornos presentan desafíos debido a factores externos como sombras y cambios en la luminosidad, el sistema en general demuestra una alta precisión y exactitud en la detección y modelado de actividades. Finalmente, el uso de SEL como lenguaje estándar para definir y comunicar actividades en sistemas distribuidos multicámara permite reducir la complejidad y fomenta la colaboración entre diversos recursos y sensores. Esto no solo unifica los elementos del sistema, sino que también abre la puerta a nuevas oportunidades para futuras investigaciones y aplicaciones prácticas en el control y análisis de actividades en diferentes escenarios.

6.1 Trabajo Futuro

El sistema de vigilancia distribuido multicámara basado en el lenguaje SEL ha demostrado ser una herramienta poderosa para modelar e inferir actividades

en diversos escenarios. A continuación, se describen algunas áreas de investigación y desarrollo que se tienen en mente para mejorar y expandir la utilidad y eficiencia de este sistema:

- **Mejora de la definición automática de estados en la escena:** Se planea investigar y desarrollar métodos avanzados para la definición automática de estados en las escenas analizadas. Esto incluiría la utilización de técnicas de procesamiento de imágenes y aprendizaje automático para identificar y categorizar automáticamente los diferentes estados de movimiento presentes en una escena. Los métodos que se deben desarrollar tienen que ser eficientes, precisos y lo suficientemente flexibles para adaptarse a los cambios en el entorno y las condiciones ambientales.

- **Generación automática de código SEL:** Para facilitar aún más la implementación y configuración del sistema, se explorará la posibilidad de desarrollar herramientas para la generación automática de código SEL. Estas herramientas permitirían configurar automáticamente el estado inicial del sistema, basándose en análisis previos y un conjunto de muestras representativas. La generación automática de código SEL facilitará una implementación más rápida y precisa en nuevos escenarios y aplicaciones, optimizando el tiempo de desarrollo y reduciendo los posibles errores humanos.

- **Optimización de métodos de detección de movimiento:** Se investigarán y desarrollarán técnicas avanzadas para mejorar la precisión y robustez de los métodos de detección de movimiento empleados en el sistema. Esto incluiría la exploración de algoritmos más sofisticados para diferenciar los objetos en movimiento del fondo, así como la adaptación de estos algoritmos para operar de manera eficiente en entornos dinámicos y complejos.

- **Integración de sensores y cámaras:** Con el avance continuo de las tecnologías de sensores y cámaras, se explorará la integración de nuevas plataformas y dispositivos en el sistema de vigilancia distribuido. Esto abarcaría

la evaluación de sensores más avanzados, como cámaras de alta resolución y sensores infrarrojos, los cuales podrían mejorar la capacidad del sistema para detectar y seguir objetos en diversas condiciones de iluminación y en entornos adversos.

- **Homogeneización y estandarización:** Para facilitar la adopción e interoperabilidad del lenguaje SEL en diversas plataformas y aplicaciones, se trabajará en el establecimiento de estándares y mejores prácticas. Esto incluirá la colaboración con la comunidad académica e industrial para desarrollar guías de implementación y documentación técnica que promuevan el uso efectivo y estandarizado del lenguaje SEL.

Referencias

Abad, C. L., Ortiz-Holguin, E., y Boza, E. F. (2021). Have we reached consensus? an analysis of distributed systems syllabi. En *Proceedings of the 52nd acm technical symposium on computer science education* (p. 1082–1088). New York, NY, USA: Association for Computing Machinery. Descargado de https://doi.org/10.1145/3408877.3432409 doi: 10.1145/3408877.3432409

Alevizos, E., Skarlatidis, A., Artikis, A., y Paliouras, G. (2017, septiembre). Probabilistic complex event recognition: A survey. *ACM Comput. Surv.*, *50*(5). Descargado de https://doi.org/10.1145/3117809 doi: 10.1145/3117809

Alferes, J. J., Banti, F., y Brogi, A. (2006). An event-condition-action logic programming language. En M. Fisher, W. van der Hoek, B. Konev, y A. Lisitsa (Eds.), *Logics in artificial intelligence* (pp. 29–42). Berlin, Heidelberg: Springer Berlin Heidelberg.

Almes, G., Black, A., Lazowska, E., y Noe, J. (1985, 02). The eden system: A technical review. *Software Engineering, IEEE Transactions on*, *SE-11*, 43 - 59. doi: 10.1109/TSE.1985.231536

Alur, R., Stanford, C., y Watson, C. (2023, jan). A robust theory of series parallel graphs. *Proc. ACM Program. Lang.*, *7*(POPL). Descargado de https://doi.org/10.1145/3571230 doi: 10.1145/3571230

Amosa, T. I., Sebastian, P., Izhar, L. I., Ibrahim, O., Ayinla, L. S., Bahashwan, A. A., ... Samaila, Y. A. (2023). Multi-camera multi-object tracking: A review of current trends and future advances. *Neurocomputing*, *552*, 126558. Descargado de

https://www.sciencedirect.com/science/article/pii/S0925231223006811 doi: https://doi.org/10.1016/j.neucom.2023.126558

Berger, J., y Lu, S. (2022). A multi-camera system for human detection and activity recognition. *Procedia CIRP*, *112*, 191-196. Descargado de https://www.sciencedirect.com/science/article/pii/S2212827122012343 (15th CIRP Conference on Intelligent Computation in ManufacturingEngineering, 14-16 July 2021) doi: https://doi.org/10.1016/j.procir.2022.09.071

Black, A., Hutchinson, N., Jul, E., Levy, H., y Carter, L. (1987, Jan). Distrbution and abstract types in emerald. *IEEE Transactions on Software Engineering*, *SE-13*(1), 65-76. doi: 10.1109/TSE.1987.232836

Chaudhary, P., Goel, S., Jain, P., Singh, M., Aggarwal, P. K., y Anupam. (2021). The astounding relationship: Middleware, frameworks, and api. En *2021 9th international conference on reliability, infocom technologies and optimization (trends and future directions) (icrito)* (p. 1-4). doi: 10.1109/ICRITO51393.2021.9596088

Cob-Parro, A. C., Losada-Gutiérrez, C., Marrón-Romera, M., Gardel-Vicente, A., y Bravo-Muñoz, I. (2021). Smart video surveillance system based on edge computing. *Sensors*, *21*(9). Descargado de https://www.mdpi.com/1424-8220/21/9/2958 doi: 10.3390/s21092958

Contreras Rivas, L. Y., López Domínguez, E., Hernández Velázquez, Y., Domínguez Isidro, S., Medina Nieto, M. A., y De La Calleja, J. (2025). A layered software architecture for the development of smart mobile distributed systems oriented to the management of emergency cases. *Applied Sciences*, *15*(7). Descargado de https://www.mdpi.com/2076-3417/15/7/3664 doi: 10.3390/app15073664

Devanne, M., Wannous, H., Berretti, S., Pala, P., Daoudi, M., y Del Bimbo, A. (2015, July). 3-d human action recognition by shape analysis of motion trajectories on riemannian manifold. *IEEE Transactions on Cybernetics*, *45*(7), 1340-1352. doi: 10.1109/TCYB.2014.2350774

Dias, H., Rocha, J., Silva, P., Leao, C., y Reis, L. P. (2005a). Distributed sur-

veillance system. En *2005 portuguese conference on artificial intelligence* (p. 257-261). doi: 10.1109/EPIA.2005.341225

Dias, H., Rocha, J., Silva, P., Leao, C., y Reis, L. P. (2005b). Distributed surveillance system. En *2005 portuguese conference on artificial intelligence* (p. 257-261). doi: 10.1109/EPIA.2005.341225

Dragon, R., Fenzi, M., Siberski, W., Rosenhahn, B., y Ostermann, J. (2012). Towards feature-based situation assessment for airport apron video surveillance. En F. Dellaert, J.-M. Frahm, M. Pollefeys, L. Leal-Taixé, y B. Rosenhahn (Eds.), *Outdoor and large-scale real-world scene analysis* (pp. 110–130). Berlin, Heidelberg: Springer Berlin Heidelberg.

Dutta, S. J., Boongoen, T., y Zwiggelaar, R. (2025). Human activity recognition: A review of deep learning-based methods. *IET Computer Vision*, *19*(1), e70003. Descargado de https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/cvi2.70003 doi: https://doi.org/10.1049/cvi2.70003

Elharrouss, O., Almaadeed, N., y Al-Maadeed, S. (2021). A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, *77*, 103116. doi: https://doi.org/10.1016/j.jvcir.2021.103116

Emmerich, W., Aoyama, M., y Sventek, J. (2008, agosto). The impact of research on the development of middleware technology. *ACM Trans. Softw. Eng. Methodol.*, *17*(4). Descargado de https://doi.org/10.1145/13487689.13487692 doi: 10.1145/13487689.13487692

Gao, M., Zheng, F., Yu, J. J. Q., Shan, C., Ding, G., y Han, J. (2023, 01 de Jan). Deep learning for video object segmentation: a review. *Artificial Intelligence Review*, *56*(1), 457-531. Descargado de https://doi.org/10.1007/s10462-022-10176-7 doi: 10.1007/s10462-022-10176-7

García-Huerta, J.-M., Jiménez-Hernández, H., Herrera-Navarro, A.-M., Hernández-Díaz, T., y Terol-Villalobos, I. (2014). Modelling dynamics with context-free grammars. En R. P. Loce, E. Saber, y N. Lecky (Eds.), *Video surveillance and transportation imaging ap-*

plications 2014 (Vol. 9026, pp. 247 – 252). SPIE. Descargado de https://doi.org/10.1117/12.2040973 doi: 10.1117/12.2040973

Gu, F., Chung, M.-H., Chignell, M., Valaee, S., Zhou, B., y Liu, X. (2021, oct). A survey on deep learning for human activity recognition. *ACM Comput. Surv.*, *54*(8). Descargado de https://doi.org/10.1145/3472290 doi: 10.1145/3472290

Jensen, K., Kristensen, L. M., y Wells, L. (2007). Coloured petri nets and cpn tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, *9*, 213-254. Descargado de https://api.semanticscholar.org/CorpusID:1180676

Karim, M., Khalid, S., Aleryani, A., Khan, J., Ullah, I., y Ali, Z. (2024, 01). Human action recognition systems: A review of the trends and state-of-the-art. *IEEE Access*, *PP*, 1-1. doi: 10.1109/ACCESS.2024.3373199

Klein, J., y van Vliet, H. (2013). A systematic review of system-of-systems architecture research. , 13–22. Descargado de https://doi.org/10.1145/2465478.2465490 doi: 10.1145/2465478.2465490

Kulsoom, F., Narejo, S., Mehmood, Z., Chaudhry, H. N., Butt, A., y Bashir, A. K. (2022, 01 de Nov). A review of machine learning-based human activity recognition for diverse applications. *Neural Computing and Applications*, *34*(21), 18289-18324. Descargado de https://doi.org/10.1007/s00521-022-07665-9 doi: 10.1007/s00521-022-07665-9

Kumar, R., y Kumar, S. (2023). Survey on artificial intelligence-based human action recognition in video sequences. *Optical Engineering*, *62*(2), 023102. Descargado de https://doi.org/10.1117/1.OE.62.2.023102 doi: 10.1117/1.OE.62.2.023102

Kumar, R., y Kumar, S. (2024). A survey on intelligent human action recognition techniques. *Multimedia Tools and Applications*, *83*, 52653–52709. Descargado de https://doi.org/10.1007/s11042-023-17529-6 doi: 10.1007/s11042-023-17529-6

LeCun, Y., Bengio, Y., y Hinton, G. (2015, 01 de May). Deep

learning. *Nature*, *521*(7553), 436-444. Descargado de https://doi.org/10.1038/nature14539 doi: 10.1038/nature14539

Liskov, B. (1988, marzo). Distributed programming in argus. *Commun. ACM*, *31*(3), 300–312. Descargado de https://doi.org/10.1145/42392.42399 doi: 10.1145/42392.42399

Meiklejohn, C., y Van Roy, P. (2015). Lasp: a language for distributed, coordination-free programming. En *Proceedings of the 17th international symposium on principles and practice of declarative programming* (p. 184–195). New York, NY, USA: Association for Computing Machinery. doi: 10.1145/2790449.2790525

Morris, B. T., y Trivedi, M. M. (2008). A survey of vision-based trajectory learning and analysis for surveillance. *IEEE Transactions on Circuits and Systems for Video Technology*, *18*(8), 1114–1127. doi: 10.1109/TCSVT.2008.927109

Morshed, M. G., Sultana, T., Alam, A., y Lee, Y.-K. (2023). Human action recognition: A taxonomy-based survey, updates, and opportunities. *Sensors*, *23*(4). Descargado de https://www.mdpi.com/1424-8220/23/4/2182 doi: 10.3390/s23042182

Nguyen, N. T., Phung, D. Q., Venkatesh, S., y Bui, H. (2005, June). Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. En *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)* (Vol. 2, p. 955-960 vol. 2). doi: 10.1109/CVPR.2005.203

Olagoke, A. S., Ibrahim, H., y Teoh, S. S. (2020). Literature survey on multi-camera system and its application. *IEEE Access*, *8*, 172892-172922. doi: 10.1109/ACCESS.2020.3024568

Pirsiavash, H., y Ramanan, D. (2014). Parsing videos of actions with segmental grammars. En *2014 ieee conference on computer vision and pattern recognition* (p. 612-619). doi: 10.1109/CVPR.2014.85

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, *28*(6), 976–990. doi: 10.1016/j.imavis.2009.11.014

Rakhimov, M., Mamadjanov, D., y Mukhiddinov, A. (2020). A high-performance parallel approach to image processing in distributed computing. En *2020 ieee 14th international conference on application of information and communication technologies (aict)* (p. 1-5). doi: 10.1109/AICT50176.2020.9368840

Ramirez-Rosales, S., Diaz-Jimenez, L.-A., Canton-Enriquez, D., Perez-Ramos, J.-L., Hernandez-Ramirez, H., Herrera-Navarro, A.-M., ... Jimenez-Hernandez, H. (2024). A state-based language for enhanced video surveillance modeling (sel). *Modelling*, *5*(2), 549–568. Descargado de https://www.mdpi.com/2673-3951/5/2/29 doi: 10.3390/modelling5020029

Ramírez, S., García, J. M., y Jiménez, H. (2018, May). Formal programming language for modeling activities using visual primitives. En *2018 xiv international engineering congress (coniin)* (p. 1-6). doi: 10.1109/CONIIN.2018.8489815

Richard, A., y Gall, J. (2016, June). Temporal action detection using a statistical language model. En *2016 ieee conference on computer vision and pattern recognition (cvpr)* (p. 3131-3140). doi: 10.1109/CVPR.2016.341

Sarray, I., Ressouche, A., Moisan, S., Rigault, J., y Gaffe, D. (2017, Oct). An activity description language for activity recognition. , 177-182. doi: 10.1109/IINTEC.2017.8325935

Shidik, G. F., Noersasongko, E., Nugraha, A., Andono, P. N., Jumanto, J., y Kusuma, E. J. (2019). A systematic review of intelligence video surveillance: Trends, techniques, frameworks, and datasets. *IEEE Access*, *7*, 170457-170473. doi: 10.1109/ACCESS.2019.2955387

Simonyan, K., y Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos.

Sobral, A., y Vacavant, A. (2014). A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, *122*, 4-21. doi: https://doi.org/10.1016/j.cviu.2013.12.005

Steen, M., y Tanenbaum, A. S. (2016, octubre). A brief introduction to

distributed systems. *Computing*, *98*(10), 967–1009. Descargado de https://doi.org/10.1007/s00607-016-0508-7 doi: 10.1007/s00607-016-0508-7

Suljkanović, A., Gostojić, S., Dejanović, I., y Kaplar, S. (2017). Analysis of current languages for developing distributed systems. Descargado de https://api.semanticscholar.org/CorpusID:66993282

Sun, Z., Ke, Q., Rahmani, H., Bennamoun, M., Wang, G., y Liu, J. (2023). Human action recognition from various data modalities: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *45*(3), 3200-3225. doi: 10.1109/TPAMI.2022.3183112

Tappenden, A., Huynh, T., Miller, J., Geras, A., y Smith, M. (2006, 01). Agile development of secure web-based applications. *IJITWE*, *1*, 1-24. doi: 10.4018/jitwe.2006040101

Tran, T. M., Vu, T. N., Vo, N. D., Nguyen, T. V., y Nguyen, K. (2022, diciembre). Anomaly analysis in images and videos: A comprehensive review. *ACM Comput. Surv.*, *55*(7). Descargado de https://doi.org/10.1145/3544014 doi: 10.1145/3544014

Valera, M., y Velastín, S. A. (2005). Intelligent distributed surveillance systems: a review.. Descargado de https://api.semanticscholar.org/CorpusID:16525678

Van Steen, M., y Tanenbaum, A. S. (2016, 01 de Oct). A brief introduction to distributed systems. *Computing*, *98*(10), 967-1009. Descargado de https://doi.org/10.1007/s00607-016-0508-7 doi: 10.1007/s00607-016-0508-7

van Steen, M., y Tanenbaum, A. S. (2023). *Distributed systems* (4th ed.). distributed-systems.net. Descargado de https://www.distributed-systems.net/index.php/books/ds4/

Vatter, J., Mayer, R., y Jacobsen, H.-A. (2023, agosto). The evolution of distributed systems for graph neural networks and their origin in graph processing and deep learning: A survey. *ACM Comput. Surv.*, *56*(1). Descargado de https://doi.org/10.1145/3597428 doi: 10.1145/3597428

Vella, G., Dimou, A., Gutierrez-Perez, D., Toti, D., Nicoletti, T., La Mattina, E., ...

Daras, P. (2021). Survant: An innovative semantics-based surveillance video archives investigation assistant. En A. Del Bimbo y cols. (Eds.), *Pattern recognition. icpr international workshops and challenges* (pp. 611–626). Cham: Springer International Publishing.

Wang, Z., Liu, S., Zhang, J., Chen, S., y Guan, Q. (2017, Aug). A spatio-temporal crf for human interaction understanding. *IEEE Transactions on Circuits and Systems for Video Technology*, *27*(8), 1647-1660. doi: 10.1109/TCSVT.2016.2539699

Yang, H. F., Cai, J., Liu, C., Ke, R., y Wang, Y. (2023). Cooperative multi-camera vehicle tracking and traffic surveillance with edge artificial intelligence and representation learning. *Transportation Research Part C: Emerging Technologies*, *148*, 103982. Descargado de https://www.sciencedirect.com/science/article/pii/S0968090X22003953 doi: https://doi.org/10.1016/j.trc.2022.103982

Yin, H., Sinnott, R. O., y Jayaputera, G. T. (2024). A survey of video-based human action recognition in team sports. *Artificial Intelligence Review*, *57*(293). Descargado de https://doi.org/10.1007/s10462-024-10934-9 doi: 10.1007/s10462-024-10934-9

Zerzour, K., y Frazier, G. (2000, 01). Vigilant: A semantic model for content and event based indexing and retrieval of surveillance video. En (p. 143-154).

Zhang, S., Li, Y., Zhang, S., Shahabi, F., Xia, S., Deng, Y., y Alshurafa, N. (2022). Deep learning in human activity recognition with wearable sensors: A review on advances. *Sensors*, *22*(4). Descargado de https://www.mdpi.com/1424-8220/22/4/1476 doi: 10.3390/s22041476

Ziaeefard, M., y Bergevin, R. (2015). Semantic human activity recognition: A literature review. *Pattern Recognition*, *48*(8), 2329-2345. doi: https://doi.org/10.1016/j.patcog.2015.03.006