International Journal of
Geo-Information

Article

Design and Development of an Internet of Smart Cameras Solution for Complex Event Detection in COVID-19 Risk Behaviour Recognition

Sepehr Honarparvar [1,2,*], Sara Saeedi [1], Steve Liang [1,3] and Jeremy Squires [3]

1 Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N 4V8, Canada; ssaeedi@ucalgary.ca (S.S.); steve.liang@ucalgary.ca (S.L.)
2 Department of Geomatics Engineering, K. N. Toosi University of Technology, Tehran 19697-64499, Iran
3 SensorUp Inc., Calgary, AB T2L2K7, Canada; jeremy.squires@sensorup.com
* Correspondence: sepehr.honarparvar@ucalgary.ca

Abstract: Emerging deep learning (DL) approaches with edge computing have enabled the automation of rich information extraction, such as complex events from camera feeds. Due to the low speed and accuracy of object detection, some objects are missed and not detected. As objects constitute simple events, missing objects result in missing simple events, thus the number of detected complex events. As the main objective of this paper, an integrated cloud and edge computing architecture was designed and developed to reduce missing simple events. To achieve this goal, we deployed multiple smart cameras (i.e., cameras which connect to the Internet and are integrated with computerised systems such as the DL unit) in order to detect complex events from multiple views. Having more simple events from multiple cameras can reduce missing simple events and increase the number of detected complex events. To evaluate the accuracy of complex event detection, the F-score of risk behaviour regarding COVID-19 spread events in video streams was used. The experimental results demonstrate that this architecture delivered 1.73 times higher accuracy in event detection than that delivered by an edge-based architecture that uses one camera. The average event detection latency for the integrated cloud and edge architecture was 1.85 times higher than that of only one camera. However, this finding was insignificant with regard to the current case study. Moreover, the accuracy of the architecture for complex event matching with more spatial and temporal relationships showed significant improvement in comparison to the edge computing scenario. Finally, complex event detection accuracy considerably depended on object detection accuracy. Regression-based models, such as you only look once (YOLO), were able to provide better accuracy than region-based models.

Keywords: internet of things; COVID-19; video streaming; complex event detection; smart camera; edge processing

check for updates

Citation: Honarparvar, S.; Saeedi, S.; Liang, S.; Squires, J. Design and Development of an Internet of Smart Cameras Solution for Complex Event Detection in COVID-19 Risk Behaviour Recognition. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 81. https://doi.org/10.3390/ijgi10020081

Academic Editors: Wolfgang Kainz and Fazlay S. Faruque

Received: 23 December 2020
Accepted: 16 February 2021
Published: 18 February 2021

1. Introduction

The final report of the National Science Foundation (NSF) workshop in 2013 predicted that there would be one camera per person in the future [1]. Cameras have always been used as low-cost equipment for gathering information about objects and events in the environment. However, monitoring and analysing events and objects in camera feeds can be challenging and, in many cases, simple events (i.e., events that do not include any event) or objects are not useful for meeting the project objectives. For example, a car driving past on another car's right is a simple event that does not generate meaningful results. However, the complex event of a car driving past from the right and overtaking at 150 km/h speed would be considered an offensive driving action. With the recent advent of video and image processing tools, complex event detection in video frames has been widely studied and applied to many domains, from anomaly [2] detection to phenomenon and behaviour tracking [3]. Complex event detection is the process of detecting complex objects based on spatial and temporal relationships in video streams [4]. Simple events are defined as the occurrence of objects in video streams [5], whilst complex events are a combination of several simple events [6]. Complex event detection has been deployed in several applications, such as transportation [7], traffic [8], agriculture [9], health monitoring [10], and abnormal behaviour detection [11]. Given its many uses, complex event detection is an important process for obtaining useful spatiotemporal information from the environment.

Some issues should be addressed in relation to using camera frames for complex event detection. One of these issues is the protection of the privacy of people whose faces and identities are revealed in camera feeds. Most of the research on object detection in camera feeds propose cryptography solutions for camera frames and significantly decreasing the object detection speed [12]. An alternative solution would be to use an architecture based on edge computing technology to provide privacy [13]. Another issue with complex event detection is that the automatic event pattern detection in frame units is challenging, as many objects and their relationships fail to go undetected. The majority of research into complex event detection has attempted to improve the querying of video content to increase the ease of information retrieval for users [14–19]. There were also other studies that focused on event pattern detection and event matching, which is the process of matching observed events frames with complex event patterns [20–22]. However, most of the research did not consider missing frames and objects for complex event detection in a real camera streaming scenario. The problem of missing objects occurs when objects exist but are not detected by the object detector. The missing frame phenomenon results from a low frame streaming rate with the edge computing because of processing latency. Missing detected objects and tracking for real automated camera frame processing pose problems for event matching. These issues sometimes occur when objects are on top of each other, or when they are not in a position in which they can be recognised by the object detector. For example, using the scenario in which we detect when a person coughs and touches a door handle, Camera1 can detect the action of a person coughing but cannot detect the door handle. However, Camera2 cannot detect the coughing action but can detect the door touching one. The deficiencies faced by both cameras can thus result in missing detected objects and tracking. Other potential reasons for missing objects in frames are low object detection accuracy and discontinuity of object tracking in consecutive frames because of the low speed of object detection and frame processing latency. In order to address such issues, in addition to the necessity of improvement to the complex event detection method, selecting a convolutional neural network (CNN) object detection model that is fast and precise is also essential.

One potential solution for this problem is gathering data from different cameras pointing to the same place. The Internet of Things (IoT) provides real-time measurements and automatically transmits information from various sources to other things. Recently, IoT and video processing technologies were integrated in order to provide smart video monitoring and system analysis [23]. This led to low-cost and high-accuracy smart monitoring systems for smart home systems [24], security border systems [25], and intelligent traffic information systems [26]. The integration of deep learning (DL) and IoT for automation in detecting and monitoring objects from video streams has been used for person re-identification using multiple cameras [27], disaster management [28], and the detection of individuals in smart surveillance systems [29]. This enables smart cameras to automatically detect objects from various camera streams and then publish them onto an Internet of smart camera (IoSC) network. For this paper, an architecture was designed and developed that aimed to decrease the occurrence of missing simple events and to increase complex event detection accuracy through the use of deep learning, the Internet of Things, edge computing, and cloud computing technologies. To address the privacy issue, the architecture should include edge computing. In order to evaluate this architecture, we considered some complex actions deemed risky in relation to COVID-19 spread in rooms within a building.

The main contributions of this paper are as follows:

- To design and develop an integrated edge and cloud computing architecture for complex event detection using the interoperable Internet of smart cameras. This flexible architecture enables us to detect objects from single cameras and integrate them into a complex event in the cloud computing part.

- To implement the integrated cloud and edge computing architecture of the IoSC to detect risk behaviours carried out by people that might lead to possible COVID-19 spread. Risk behaviours are considered complex events and are detected using the architecture model and reported to the cloud network in real-time.
- To demonstrate how much the object detection model influences the number of missing events, we compared region- and regression-based object detection models for simple events.

The remainder of this paper is organised as follows: Section 2 reviews work related to complex event detection related work, whilst Section 3 explains the architecture details; Section 4 gives an overview of the implemented architecture, and Section 5 discusses the results of the implemented system; the final section then sums up the findings for this paper and provides suggestions for future research.

2. Background

One of the most interesting examples of event detection in academia involves human actions and the behaviours of people. The application of DL in video surveillance for behaviour recognition and action detection has attracted the attention of scholars. Behaviour recognition is often defined as the classification of a specific human behavioural patterns in video streams. Most of the studies in this field have attempted to extract motions by using optical flow analysis [30]. Optical flow analysis is used for tracking objects through consecutive frames. The other research category for behaviour recognition focuses on building innovative CNN for feature extraction and classification from frames. Building a 3D CNN for action recognition is an example of these efforts [31–33]. CNN is based on spatiotemporal training for object motion detection. However, the results of these models were only able to achieve results with a high level of accuracy by using standard video streaming. Another approach researchers used was considering behaviours as objects and trying to detect the objects in real-time in the video frames [34]. Ko and Sim proposed a DL framework to discriminate and detect abnormal behaviours in order to address the issue raised by previous work on discriminating human behaviours in a crowd scene [35]. However, these studies only considered human behaviours as potential events and did not detect relationships between events in time or space.

Complex event detection is more challenging than behaviour detection as the events are more spatially and temporally complex [4]. Some of the research used DL in video streams for complex event detection by relating simple events. Most of the research focused on the classification of videos based on particular classes using DL. Gan et al. developed a DL model to extract keyframes for predefined complex events [36]. Xiong et al. used DL to detect people–object interactions as events [37]. Several papers such as [38] and [39] performed classification using DL and multimodal information. These studies utilised the integration of multimedia DL models in order to provide a better estimation of events in the real world. Other studies, such as [40–42], used a concept detector to detect events in video streams. However, these studies are only relevant to complex events occurring within a short period.

The most current existing complex event detection software and frameworks on the market, such as Oracle complex event processing (CEP), WSO2 CEP, ApacheFlink CEP, and StreamInsight, are mainly designed to detect only complex events and temporal relationships of events whilst neglecting spatial relationships. For example, Medioni et al. detected events by detecting, tracking, and extracting moving object trajectories, speed estimation, and the inference of object behaviours from frames. This study had spatial and temporal resolution constraints as it was based on pixel-wise detection. Dubba et al. designed the relational event model induction (REMIND), which uses a supervised inductive relational learning technique to detect events in a large video dataset [43]. Studies such as NoScope [44] and Focus [45] attempted to optimise spatiotemporal querying in video streams but did not provide expressive querying for retrieving spatiotemporal patterns. Hence, Yadav and Curry proposed a video complex event framework that detected spatiotemporal patterns in video streams [20]. Additionally, they proposed a query language that processed complex events from spatial and temporal relationships between objects. However, none of the above research focused on missing spatiotemporal relationships in a real scenario in which object detection misses objects in video streams. Furthermore, they all proposed single-source complex event detectors for offline video streams when all frames are available. For this paper, we designed and developed an integrated edge and cloud computing architecture to process complex events from multiple cameras in order to solve the problem of missing object detection. Other research, such as EdgeCEP [46], used cloud and edge computing for complex event detection but did not provide any solution for video complex event detection when the events were from different sources.

3. Complex Event Detection Methodology

This paper's main aim is to explain the integrated edge and cloud computing architecture components and relationships in detail. Figure 1 shows the architecture, which was divided into sensing, edge computing, and cloud computing. The edge computing part detects and tracks objects, detects simple and complex events, and publishes them using the cloud computing part. The camera feeds are first read one after another; then, the edge computing part of the architecture uses object DL detection algorithms to detect the objects that are included in the complex event model. This step results in a vector of detected object bounding box coordinates, class names, and confidence scores. The next step is to analyse consecutive frames, and track, identify, and label every detected object. Every object has a global ID in all of the frames of the detected object vector. The object detection vector and the object tracking vector are sent to an event matching process using edge computing. Spatial relationships are then extracted using information from the detected object vector. In addition, the tracked object vector is deployed to identify temporal relationships between objects in consecutive frames. To match the extracted spatiotemporal relationships of objects with existing event patterns, a query is sent to the edge event database to find similar complex events. Finally, the detected event is formatted and sent to the cloud as detected objects. Edge computing is the only component of the architecture that can access video frames. This component detects objects and only returns object bounding boxes, class, and tracking ID to the cloud components. Therefore, objects and photos of people are not distributed and/or saved anywhere. Thus, this guarantees the privacy of people and objects in video streams.

In particular, the cloud computing part is responsible for detecting complex events that have not yet been detected because of missing objects in the frames. This means that if the edge computing part fails to detect complex events from one camera, the cloud computing part makes up for this by detecting some of the missing complex events from different cameras. Therefore, other camera detection results, including time and the detection vector, are needed to find the spatial and temporal relationships and to finally detect the complex events. Firstly, detected object vectors and tracked object vectors are sent to both the cloud event matching processing service and the cloud database. The service is connected to the cloud event database, which is similar to the edge event database and includes event spatiotemporal patterns. After obtaining the detection and tracking vector from the edge computing, the cloud matching service sends queries to the cloud database to search for previously used detection and tracking vectors. The results for matched events are then sent to the cloud database.

The following subsections provide more details about object detection and tracking, event matching with edge computing, and event matching in the cloud, which are the most important units of this architecture.

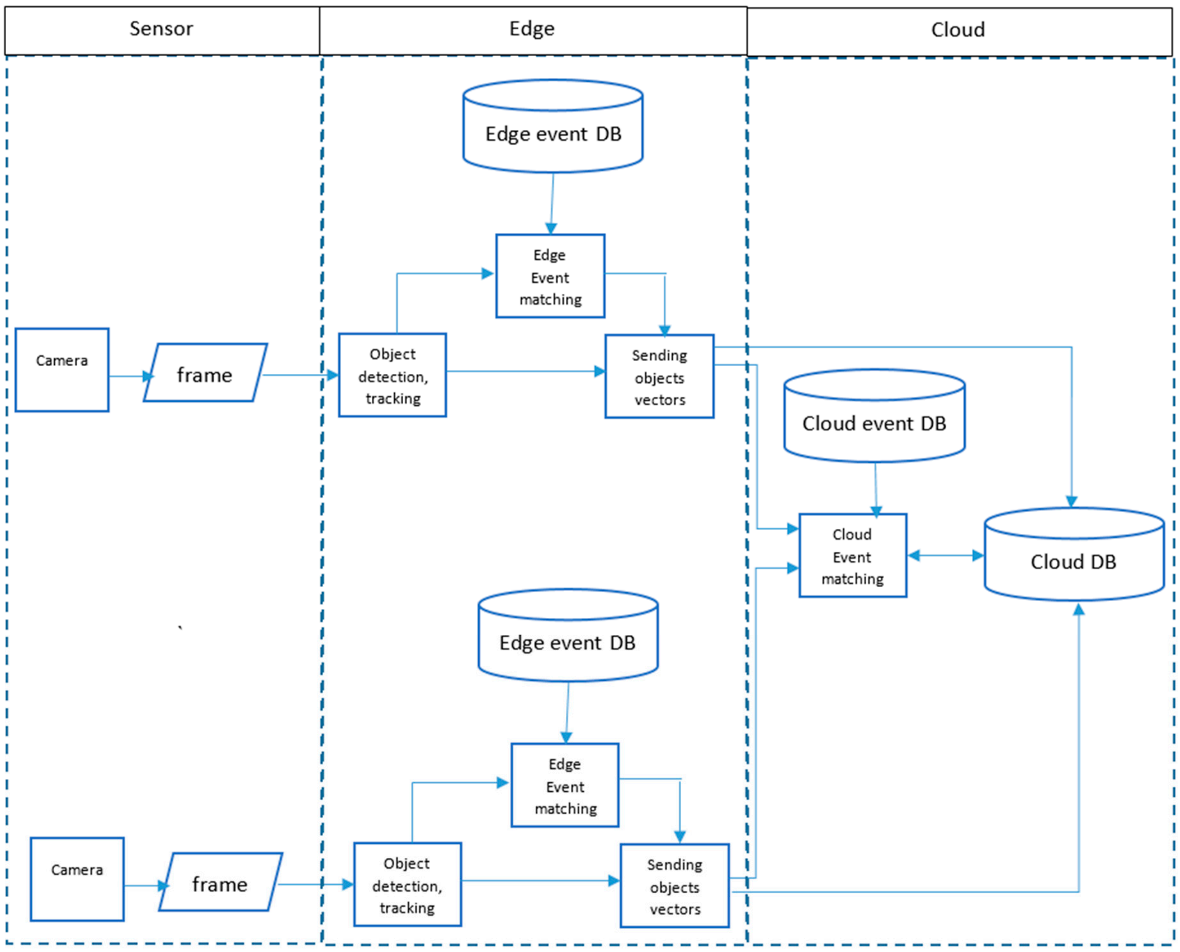


Figure 1. The designed integrated architecture for complex event detection (DB: DataBase).

3.1. *Object Detection and Tracking*

Simple events are generated in every frame, and object detection is used as the first step in extracting events from video frames. The designed architecture uses DL algorithms to automate object detection from video streams. Deep neural networks (DNNs) offer an in-depth architecture model for learning complex objects using CNN-based models [3]. Regional-based CNN and you only look once (YOLO) (regression-based model) are examples of such models. YOLO divides the image into small grids and predicts the objects in every grid before assigning a confidence score to each object. YOLO is able to detect objects faster than regional-based CNN (R-CNN) models [47]. However, it provides less accuracy, especially with regard to small objects [48]. For R-CNN models, the regions of proposals are extracted, CNN features for each region are computed after wrapping the regions, and then the regions are classified [47]. The process of preparing and building an object detection model is similar. Firstly, images are labelled based on the model-required format in the two datasets used for training and testing. Then, using transfer learning, general layers are borrowed from a pre-trained model and only the last layers are changed. This overcomes the limitations of semi-supervised solutions as well as scarcity problems for the dataset [49]. The output of this process is an object detection model. Extra information about the precision or recall of the model based on the test dataset is also included. The output model is deployed as a predictor for real-time detection.

Object tracking is the process of finding similar objects in video frames and identifying them. It is an essential part of complex event detection and identifies objects in consecutive frames to enable the detection of temporal relationships between events in the frames. There are different methods for tracking objects in video frames. Advancements in object detection have made tracking by detection one of the most popular tracking approaches. Deep simple online and real-time tracking (DeepSORT) is one of the algorithms used in vehicle tracking [50], pedestrian tracking [51], ball tracking [52], etc. The algorithm only considers the previous and current frames in order to assign an ID to the detected object. It does not need to process the entire video every single time. The objective is to minimise the following cost function [53]:

$$C_{ij} = \lambda d^{(1)}(i,j) + (1-\lambda)d^{(2)}(i,j) \tag{1}$$

where $d^{(1)}$ is the spatial distance between the predicted bounding box from Kalman filtering (KF) and the previously known bounding box for the object. The visual distance $d^{(2)}$ represents the smallest cosine distance between the bounding boxes of the matched objects in consecutive frames. i and j represent two objects in different frames. λ determines the influence of distances in the tracking process.

3.2. Event Matching with Edge

Event matching is the process of matching the registered event model in the event database with the spatiotemporal pattern in the video. The entire process of event matching for this part of the architecture is carried out with edge computing. As all of the frames are coming from a single source, the coordinate system and camera angle are the same during the complex event detection process. The process consists of object matching, spatial matching, and temporal matching [20]. Pseudocode 1 in Appendix A shows the steps for our proposed event matching algorithm.

Sample spatial and temporal queries are shown in Table 1, which illustrates direction-based, topology-based, and geometric representations of objects as spatial relationships. The sequence (SEQ) temporal relationship determines the order of events. If two spatial relationships occur within a frame, then these are equal (EQ) temporal relationships. Conjunction (CONJ) and disjunction (DISJ) refer to AND and OR logical operators. Video event query language (VEQL) [19] was used for querying objects in the architecture. For the first example, the door-touching spatial relationship considers the intersection of the hand with the door handle in a time threshold of 10 frames with the least average confidence of 0.5 for object detection. The second temporal relationship considers the sequence of coughing spatial relationships before the door-touching spatial relationship in a time threshold of 3600 frames with the least average confidence of 0.5 for object detection.

Table 1. Sample spatial/temporal relationships and queries.

Item	Spatial	Temporal
Relation	Direction-Based Spatial Relation (FORS) = (left,right,front,back)	SEQ $\rightarrow$ E1.t1 < E2.t2 where t1 < t2
Relation	Topology-Based Spatial Relation = 9Intersection Matrix	EQ $\rightarrow$ E1.t1 = E2.t2
Relation	Geometric Representation for Spatial Object (O) = e.g., Overlay Area of Bbox	CONJ $\rightarrow$ E1 and E2DISJ $\rightarrow$ E1 or E2
VEQL Query Examples	SELECT intersection (Object1, Object2) FROM Camera1 WHERE Object1.label = 'hand' AND Object2.label = 'door handle' AND WITHIN TIMEFRAME_WINDOW(10) WITH_CONFIDENCE > 0.5	SELECT SEQ (sr1, sr2) FROM Camera1 WHERE sr1.label = 'coughing' AND sr2.label = 'door touching' WITHIN TIMEFRAME_WINDOW(3600) WITH_CONFIDENCE > 0.5

For Pseudocode 1, the capureTime function obtains the time that the frame is captured. The object detector function returns all of the detection object vectors in the frame and will skip that frame if no object is found. Otherwise, it runs events from the event patterns array. The eventPatterns array is obtained from the edge event database. The object event matcher then executes the array in order to return the object events to the frame found in the event vector. If any of the detected objects matched with the object nodes from the event, it will then operate the spatial event matcher. Every spatial relationship involves two objects. Hence, the find function is run to find detected objects in the frame that are involved with the spatial relationship sr. Having run the spatial relation function between the two found objects (i.e., obj1 and obj2), it then checks if the relationship is true. If so, the spatial_rel in the frame is stored as detected spatial relationships. Similarly, the temporalRelation checks whether the temporal relationship is valid. Finally, root relationships between temporal events are matched in the last layer. If any of the rootRelationship values are false, it will skip the frame and check the next frame. The output of the algorithm is the reported complex event and the involved temporal relationship vectors.

3.3. Event Matching Using Cloud Computing

Event matching in the cloud almost always follows the event matching instructions from edge computing. The spatiotemporal queries are similar to event matching with the edge. However, as object vectors come from different cameras with different viewpoints, more steps should be taken. Object detection and tracking are requirements for running event matching. If the cloud computing services are responsible for the operations, the raw frames have to be sent to the cloud for object detection and tracking. This reduces the entire system performance and increases the network's traffic load. To avoid this, we have to find a solution that detects the spatiotemporal relationships based on the detected object vector.

The bounding boxes for detected objects come from different local image coordinate systems. A consistent coordinate system is required for all of the reported bounding boxes in order to find the spatial relationships between objects. For this paper, we use projective transformation to transform the image coordinate system into a geographic coordinate system [53–56]. Coordinate transformation allows the IoSC to project detected objects from different cameras onto a common coordinate system. If the projected objects are from the same class, the IoSC checks the intersection of these projected objects. If two objects intersect, the IoSC matches them and identifies them as the same object. If more than two objects are found, the IoSC selects the two objects with the most intersection and proximity for consideration. We used homomorphic (2D projective) transformation in our research. For trackable objects such as persons, the IoSC projects all objects to ground plane. As we need to project all points onto a common plane, the middle of the lower edge of bounding boxes of objects is projected to the ground. Then, the projected object coordinates are transformed to the other camera coordinate system in order to identify the tracked object from Camera2. To match other objects and find identical objects taken by two cameras, they are projected onto a wall plane (i.e., a plane which is perpendicular to the ground plane and within the line of sight of the camera). Therefore, having at least four control points for each camera transforms all of the bounding box coordinates onto the common coordinate system. This enables the cloud computing service to track objects from different cameras located within a unique coordinate system, thus making the extraction of spatiotemporal relationships possible.

Figure 2 presents an overview of the event matching flow in the cloud. After reading a frame from Camera1, the detected objects are sent to the cloud database and cloud computing service. Firstly, relevant cameras (i.e., cameras that have overlying scenes) are queried in order to obtain the eight transformation parameters. Then, the projective transformation is run in order to convert all of the detect object vectors over to the Camera2 coordinate system. The object detected vector values are queried based on their detected object class and the phenomenon time (i.e., the frame read time). The spatial and temporal relationships are then extracted using the tracking vector for the current detected object and queries from the cloud database. Finally, the event matching module detects the events and reports them to the cloud database. A sample video in Supplementary Materials explains how two cameras with a common area of visibility can complete each other when a cloud computing approach is used.

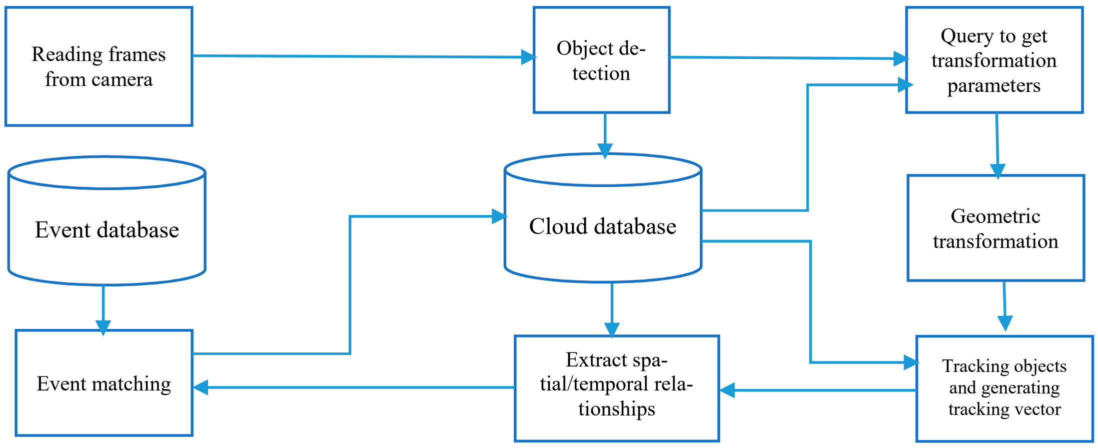

Reading frames from camera
Object detection
Query to get transformation parameters
Event database
Cloud database
Geometric transformation
Event matching
Extract spatial/temporal relationships
Tracking objects and generating tracking vector

Figure 2. Event matching steps in the cloud.

4. Implementation

COVID-19 is a global disease that has affected over 185 countries and has impacted jobs, social life, and the economy. Tracking risk behaviours displayed by people with regard to virus spread can help to prevent increasing numbers of COVID-19 cases. Recent research used radiofrequency technology to capture human motion and detect coughing in a room [57]. This approach requires special devices and infrastructure setup. In our recent study, we instead attempted to use low-cost camera devices to detect simple risk behaviours such as coughing and high population density in rooms to estimate the impact of risk behaviours on the spread of COVID-19 [58]. However, in many cases, these behaviours are complex and can include several relevant actions carried out within a certain time period. Therefore, a possible solution would be to model these behaviours using complex events. In order to constitute these events from simple events, objects are automatically detected and tracked using smart cameras and DL techniques. Complex events are then detected using a complex event detection process. For this paper, we used COVID-19 risk action detection as a use case in order to implement and evaluate the architecture.

4.1. Implemented Architecture

Figure 3 shows the detailed architecture we developed to evaluate our proposed method. For edge computing, we used the Jetson Xavier NX board with a GPU processor with 384 NVIDIA CUDA cores, 48 Tensor cores, and a 6-core NVIDIA Carmel ARMx64 CPU. A USB camera is connected to the board to record video streams. Jetson Xavier reads frames, detects the objects in each, and then sends the information as a message queuing telemetry transport (MQTT) message to the amazon web service (AWS) IoT Core. There are three topics for the message. One is for triggering the SensorThings application programming interface (STA) formatting lambda function, another for triggering the event matcher lambda function, and the last one for updating the dynamoDB database. The dynamoDB database includes an event patterns table in javascript object notation (JSON) format and a detected event vector. JSON simplifies the representation of data [59], makes hierarchical data suitable for storage [60], and is adopted for not only SQL (NoSQL) databases and different programming languages [61]. After the event matcher is triggered, it selects both registered event patterns from the dynamoDB as well as recorded detected objects from the STA endpoints. Then, the Lambda function runs the cloud event matcher to match events. When a new event is reported to the IoT core, the dynamoDB updating rule is triggered to update the event vector table. At the same time, Jetson Xavier also sends the detected object vectors to the IoTCore. As soon as the message is published to the IoTCore, the STA formatting Lambda function converts the MQTT messages into STA format and sends them to STA endpoints.

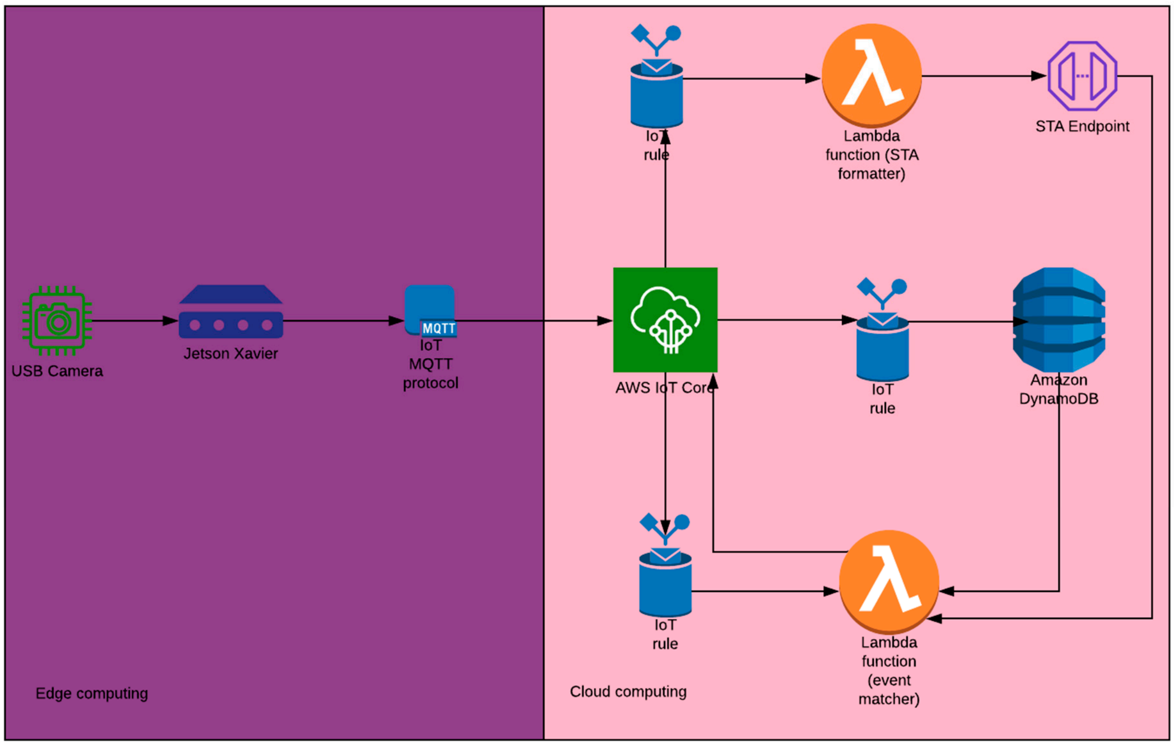
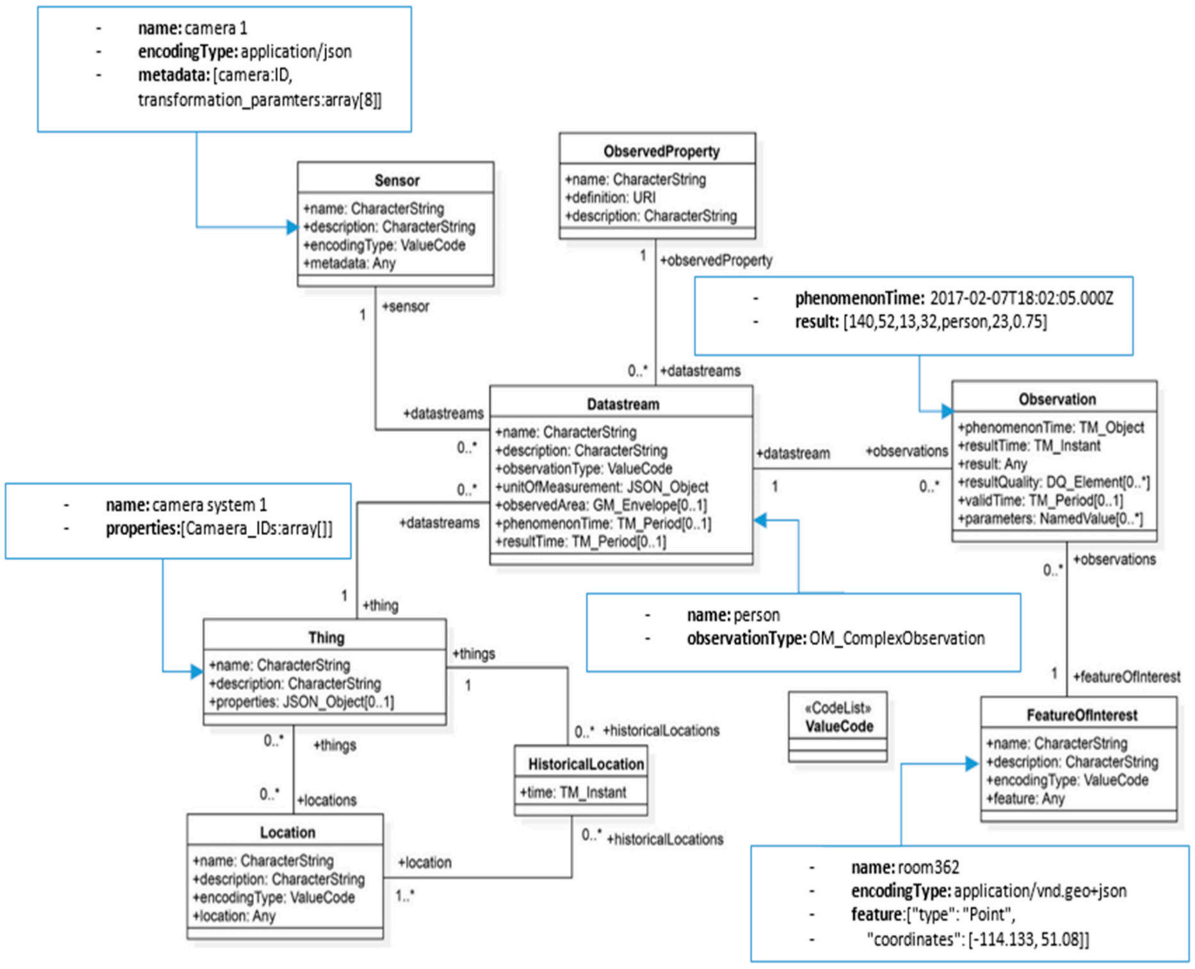


Figure 3. Implementation diagram of the developed architecture (IoT: internet of things, STA: SensorThing API, MQTT: message queuing telemetry transport, AWS: Amazon web services).

4.2. STA Model

The developed architecture involves multi-connected cameras. Therefore, a comprehensive data model is essential for modeling devices and data component relationships. The STA is a web-based open standard that interconnects devices, data, and applications. It is a geospatial-enabled Open Geospatial Consortium (OGC) standard that supports both MQTT and HTTP standards [62]. STA was used to implement the IoSC so as to provide interoperability between smart cameras. Figure 4 shows the STA data model. The black boxes are the STA entities defined by OGC [63], and the blue boxes are the examples that we used in our model. For our proposed data model, every camera act as a sensor. Related cameras and the relevant projective transformation parameter values are attributed by this entity's metadata. The relevant cameras constitute a thing. Datastreams are detected object data types based on the class of the object. For example, there is a datastream for the person and another datastream for the door. Observations contain a vector of bounding boxes, class names, tracking ID of the detected object, and the confidence score. Finally, FeatureofInterest determines the area that the cameras record. Using this structure, we can take advantage of relational queries to obtain camera transformations and detected object observations. All camera locations and FeatureofInterests are stored in GeoJSON format. GeoJSON encodes geographic data in JSON format [64]. It is a simple and lightweight data format, which is adapted to work with many map libraries and services, such as leaflets [65], OpenLayers, MapBox, and Cesium. It also supports different geographic coordinate systems and features [66].



- name: camera 1
- encodingType: application/json
- metadata: [camera:ID, transformation_paramters:array[8]]
Sensor
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+metadata: Any
ObservedProperty
+name: CharacterString
+definition: URI
+description: CharacterString
+observedProperty
+sensor
- phenomenonTime: 2017-02-07T18:02:05.000Z
- result: [140,52,13,32,person,23,0.75]
+datastreams
Datastream
+name: CharacterString
+description: CharacterString
+observationType: ValueCode
+unitOfMeasurement: JSON_Object
+observedArea: GM_Envelope[0..1]
+phenomenonTime: TM_Period[0..1]
+resultTime: TM_Period[0..1]
Observation
+phenomenonTime: TM_Object
+resultTime: TM_Instant
+result: Any
+resultQuality: DQ_Element[0..*]
+validTime: TM_Period[0..1]
+parameters: NamedValue[0..*]
+datastream
+observations
- name: camera system 1
- properties:[Camaera_IDs:array[]]
+thing
- name: person
- observationType: OM_ComplexObservation
Thing
+name: CharacterString
+description: CharacterString
+properties: JSON_Object[0..1]
+things
+historicalLocations
+featureOfInterest
«CodeList»
ValueCode
FeatureOfInterest
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+feature: Any
HistoricalLocation
+time: TM_Instant
+locations
Location
+name: CharacterString
+description: CharacterString
+encodingType: ValueCode
+location: Any
+location
- name: room362
- encodingType: application/vnd.geo+json
- feature:["type": "Point",
- "coordinates": [-114.133, 51.08]]

Figure 4. Proposed data model based on SensorThings API.

Two types of database tables have been used for this architecture. The first one includes predefined event patterns as well as spatial and temporal relationships between objects in different frames in an event pattern database. In general, each record from this table is a tree of events and related objects in different frames. The second type of database table is the cloud database table. Cloud database tables are comprised of cameras, detected object vector boxes, tracked object vectors, and matched events. Figure 5 is a logical model of a simplified schematic cloud database showing different required tables for cloud computing and storing events.

The proposed cloud database model has a metadata table containing camera information for event matching. Every detected object in the detection vector table has a camera ID connecting objects to a camera in the camera metadata table. The related camera IDs determine if cameras point to the same area (or if the areas overlap). This is used for the event matching of data from different cameras in the cloud. The array of control points is used for the frame coordinate transformation between cameras, as object locations are not the same for different cameras with different viewpoints. The detection vector includes tracked and detected vectors. If an object has not been identified in the image, the TrackID is set as NULL. This ID is the unique ID of detected objects. The confidence score is the generated probability of correct object detection by the object detection model. The matched event table stores detected events with the reported time interval, IDs of involved objects, and a Boolean attribute that specifies whether events are detected in the edge computing or not.

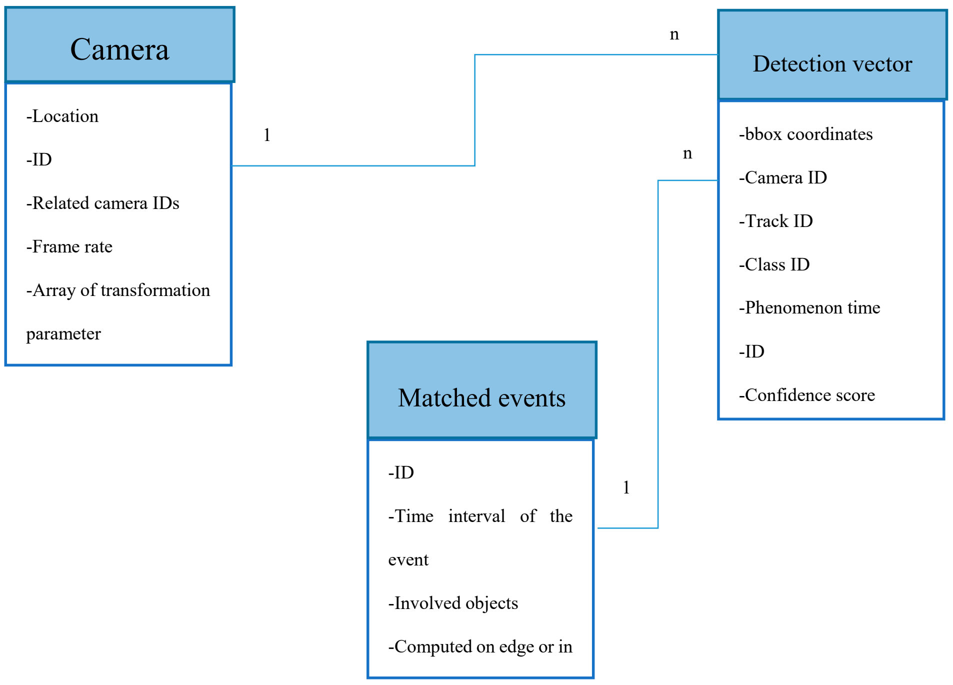


Figure 5. Logical model for the cloud database.

4.3. Data and Object Detection Model

A total of 4784 images from various sources were used for training the object detection DL model. These images were used to train the DL model for arm_coughing, hand_coughing, person, hand, door, and door handle. For training the DL model for arm_coughing and hand_coughing, we retrieved public images from Google and manually labelled them. Person-labelled data were obtained from the CoCo dataset [67]. The hand dataset [68] was prepared by the Visual Geometry Group of the University of Oxford. The door and door handle dataset comes from MCIndoor20000 [69] and Open Images Dataset V4 [70]. In order to train the detection model, two of the most popular CNN object detection techniques, Mask-RCNN [71] (region-based) and Yolov3 [47] (representative regression-based method), were deployed. For both models, the CoCo pre-trained model was used for transfer learning.

Figure 6 shows the graph regarding complex events of risk behaviours for the spatial/temporal and object relationships. This event occurs when a person coughs into his/her arms or hands and then moves and touches a door handle. As coughing into the arm versus hands produces different risk values, we can consider them as different complex events, although the general event structure is similar. The lowest level of the graph shows objects which are detected by a CNN object detection model. The second level boxes show the spatial relationships between the objects. Intersected bounding boxes were identified in order to extract the related objects in the frame. As we required a door handle for a door and not a handle for a drawer, door and door handle bounding boxes had to intersected to ensure verification of the correct door handle object. Temporal relationships (sequence relationships that consider the order of events) were then depicted in the third level. For root relationships, the relationship between temporal relationships was considered before the complex event was finally detected. VEQL proposed by [20] was used for querying events.

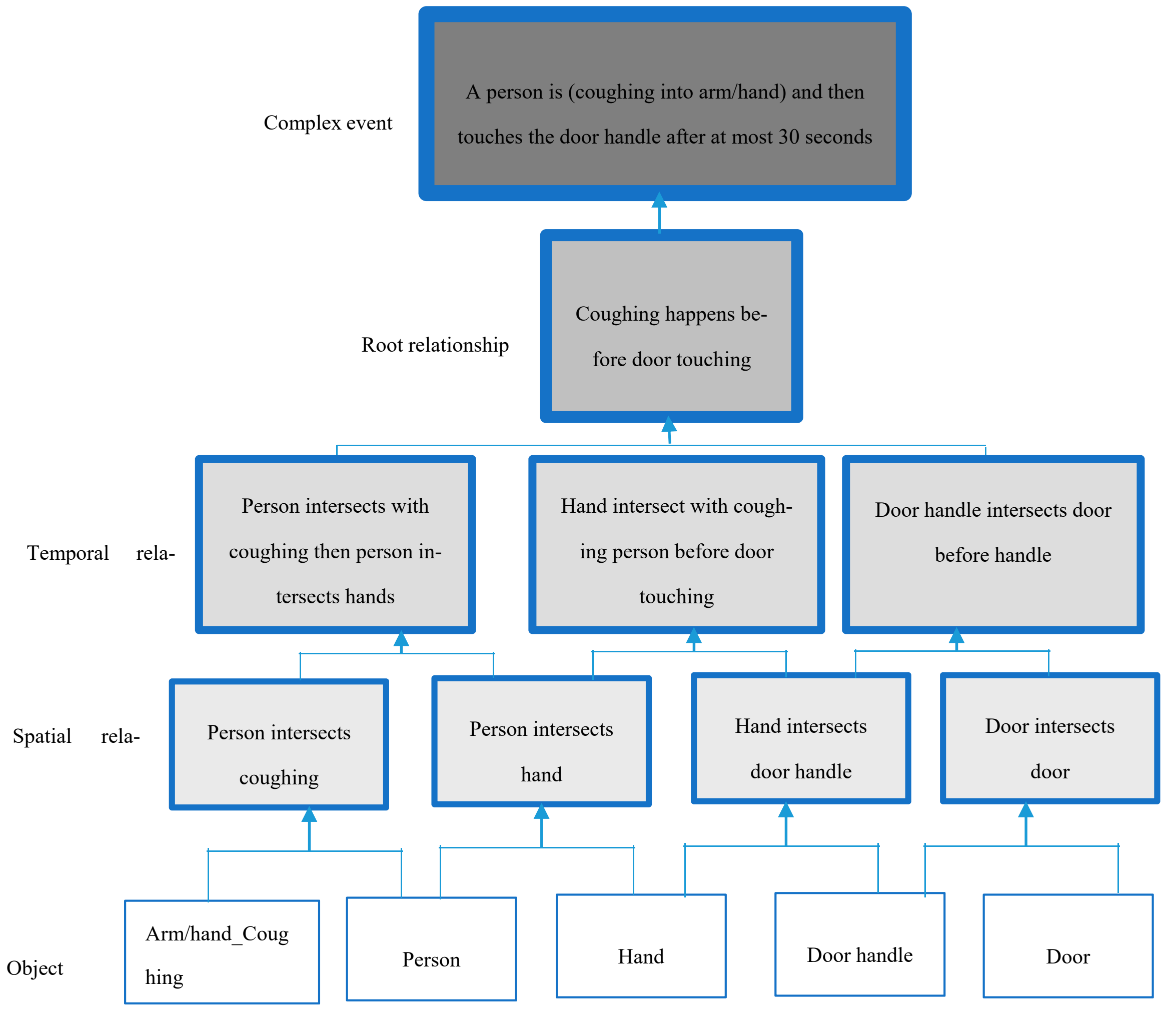

Complex event
A person is (coughing into arm/hand) and then touches the door handle after at most 30 seconds
Root relationship
Coughing happens before door touching
Temporal rela-
Person intersects with coughing then person intersects hands
Hand intersect with coughing person before door touching
Door handle intersects door before handle
Spatial rela-
Person intersects coughing
Person intersects hand
Hand intersects door handle
Door intersects door
Object
Arm/hand_Coughing
Person
Hand
Door handle
Door

Figure 6. Spatial, temporal, and object relationships graph for risky behaviours.

The output of every detected complex event is a GeoJSON record that is stored in a DynamoDB table. The GeoJSON file includes the trajectory (in line-string format) of the complex event in the geographic space as well as properties containing the phenomenon time, results time, duration, JSON format of relevant objects, JSON format of related spatial and temporal relationships, and an array of related camera names.

5. Experimental Results

To evaluate the developed architecture, the object detection models, event matching accuracy, and performance of the complex event detection were assessed.

5.1. Object Detection Accuracy

As the accuracy of object detection models affects the complex event detection accuracy and performance, two popular object detection models, Yolov3 and Mask-RCNN, were deployed for this method. To evaluate the results of object detection, two videos with 612 and 518 frames, respectively, were selected. Detections in all of the tests with a confidence score of more than 85% were considered. Figures 7 and 8 provide an overview of the tested video results.

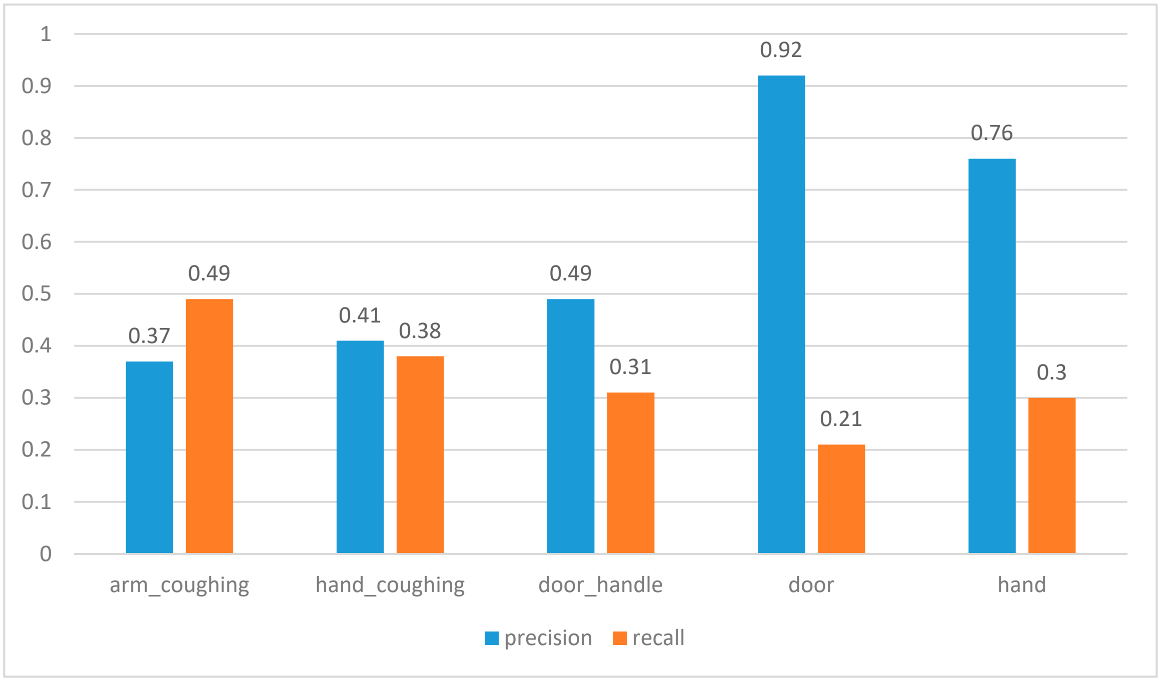


Figure 7. Object detection results of YOu Look at Once (YOLO) architecture.

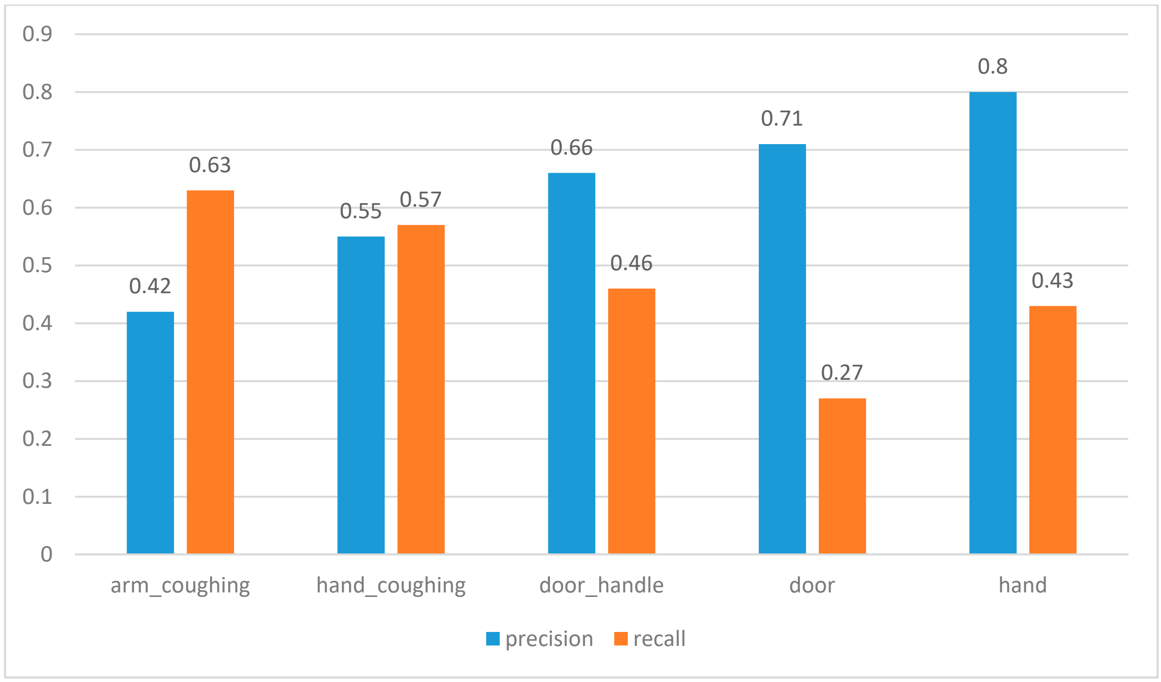


Figure 8. Object detection results of Masked regional convolutional neural network (Mask-RCNN) architecture.

When taking classes into consideration, we found that for less complicated objects, such as "door", YOLO generated better results. In comparison, Mask-RCNN generated more true positives for every class. Table 2 provides an overview of the performance and precision of the object detection models using Jetson Xavier and a laptop with Core i7 CPU and GeForce RTX 2070 GPU.

The Mask-RCNN model provided greater mean average precision (mAP) and average recall than YOLO. However, this difference is insignificant. In comparison, Yolov3 offered faster detection speeds. The number of missing objects for the Yolov3 model was fewer than that of Mask-RCNN. In general, although its precision is lower than the Mask-RCNN for complex objects, Yolov3 is a better option for complex event detection. Dividing and detecting complex objects using a Yolov3 model for simple objects in addition to considering the spatial relationships for building complex objects is probably the best approach. As a result, the deployed object detection CNN model used for the two evaluations was Yolov3. In general, the calculated mAP values were fewer than those of popular models, such as CoCo, because the training dataset size was smaller.

Table 2. YOLO and Mask-RCNN comparison results.

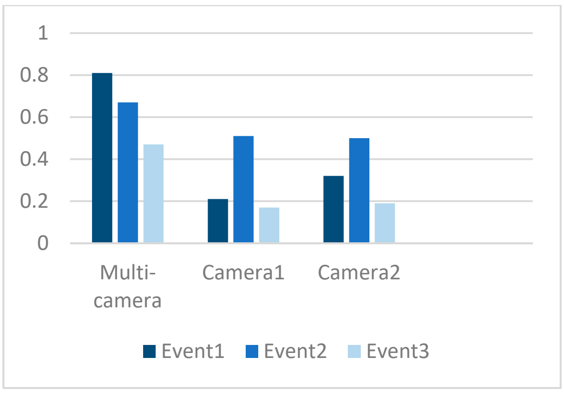
Model	Speed (Jetson Xavier)	Speed (Laptop)	Missing Objects	mAP	Average Recall
YOLOv3	4 (fps)	25 (fps)	23%	59%	34%
Mask-RCNN	0.6 (fps)	6 (fps)	37%	62%	47%

5.2. Online Event Matching Accuracy

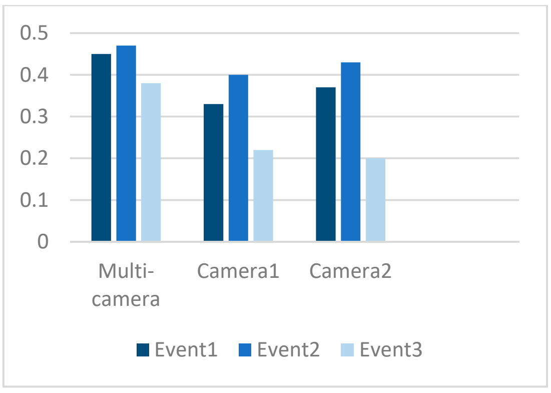
The accuracy of event matching for the integrated edge and cloud computing architecture compared to an architecture that handles all of the processes with edge computing using images from a single camera will determine the architecture event matching accuracy. The event matching accuracy was calculated based on the F-Score value (Equation (3)). As it is a metric which involves both precision and recall, it provides more comprehensive information about complex event detection.

$$F - score = \frac{2 \times precision \times recall}{precision + recall}. \tag{2}$$

Therefore, F-Scores were calculated for three complex events in two videos. Two different scenarios for complex event detection were recorded by two cameras. The average values from the two cameras for each scenario were considered for both the edge computing and offline scenarios. Video 1 is the first video with 3812 frames, including 17 complex events, and Video 2 is the second video with 4218 frames, including 23 complex events. Figure 9 provides the precision and recall values for all three events in the two videos. In this Figure, Event 1 represents coughing into hands and then touching the door handle (six occurrences in Video 1, seven occurrences in Video 2); Event 2 is coughing into arms and then touching the door handle (four occurrences in Video 1, seven occurrences in Video 2); finally, in Event 3, a person coughs into his/her hands and then arms, and then touches the door handle (seven occurrences in Video 1 and nine occurrences in Video 2). All of the values in Figure 9 are the average values of precision and recall for two videos. Values of recalls increased significantly when using multi-camera architecture, although its precision increase is insignificant. This is due to the proposed architecture reducing the number of false-negative events, which, in doing so, significantly improved the recall values.



(**a**)



(**b**)

Figure 9. Complex event values: (**a**) Recall; (**b**) precision.

Figure 10 presents the F-score results for this architecture and for the single-camera approach for Camera1 and Camera2. In this Figure, the average F-Score values from two videos are the used events. A more detailed figure is available in in Appendix B.

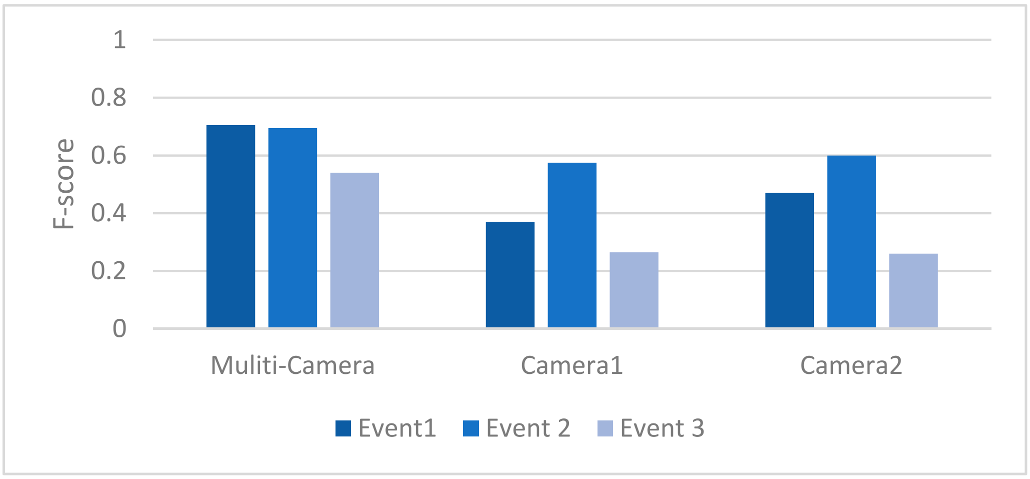


Figure 10. Event matching F-score results for the integrated architecture with multi-cameras, Camera1, and Camera2 for three events.

This overview of F-scores demonstrated that the accuracy of complex event detection with the integrated architecture was about 1.73 times higher than that of the edge computing scenario. This was even more significant in the case of more complex events (e.g., Event 3). This occurred because the integrated architecture generates more true positives (TPs) and fewer false negatives (FNs) in object detection. Finally, as expected, object detection accuracy affected the final complex event accuracy. For instance, Event 1 (coughing into hands) resulted in higher complex event accuracy than Event 2 (coughing into arms), because the average precision of coughing into hands is greater than the object class for coughing into arms. Events which include more simple events tend to result in more errors being made, as errors for each simple event affects the final complex event accuracy. Consequently, the F-Score for Event 3 is lower than for Event 1 and Event 2 for all complex event detection scenarios because Event 3 involved more simple events and spatial–temporal relationships.

5.3. Online Event Detection Speed

In order to evaluate the speed of the developed architecture, the time latency of event matching was considered for offline event detection using a laptop and for online event detection with Jetson Xavier using single-camera processing as well as the integrated architecture. We used Equation (2) to calculate relative latency. In the equation, t_e is the time that the cloud database was notified, t_f is the frame reading time, and t_w is the time that was defined as the threshold (e.g., coughing and then touching the door handle should take less than two minutes in order for it to be considered a risk behaviour) for the event in the event pattern database.

$$l = \frac{t_e - t_f - t_w}{t_w}. \tag{3}$$

In Figure 11, the average time latency of the three events with different t_w for the three scenarios was presented. The processing speed for object detection for an offline scenario was 25 fps, while it was around 4 fps for the edge computing scenario. The tw for Events 1 and 2 was 30 s, while it was one minute for Event 3. The illustrated values represent the average latency of all of the cameras and videos. Therefore, for the offline and edge computing architecture, the values were the average of the average of the latency of the two cameras in the two scenarios. The integrated architecture latency values are the results of averaging the latency of Scenarios 1 and 2.

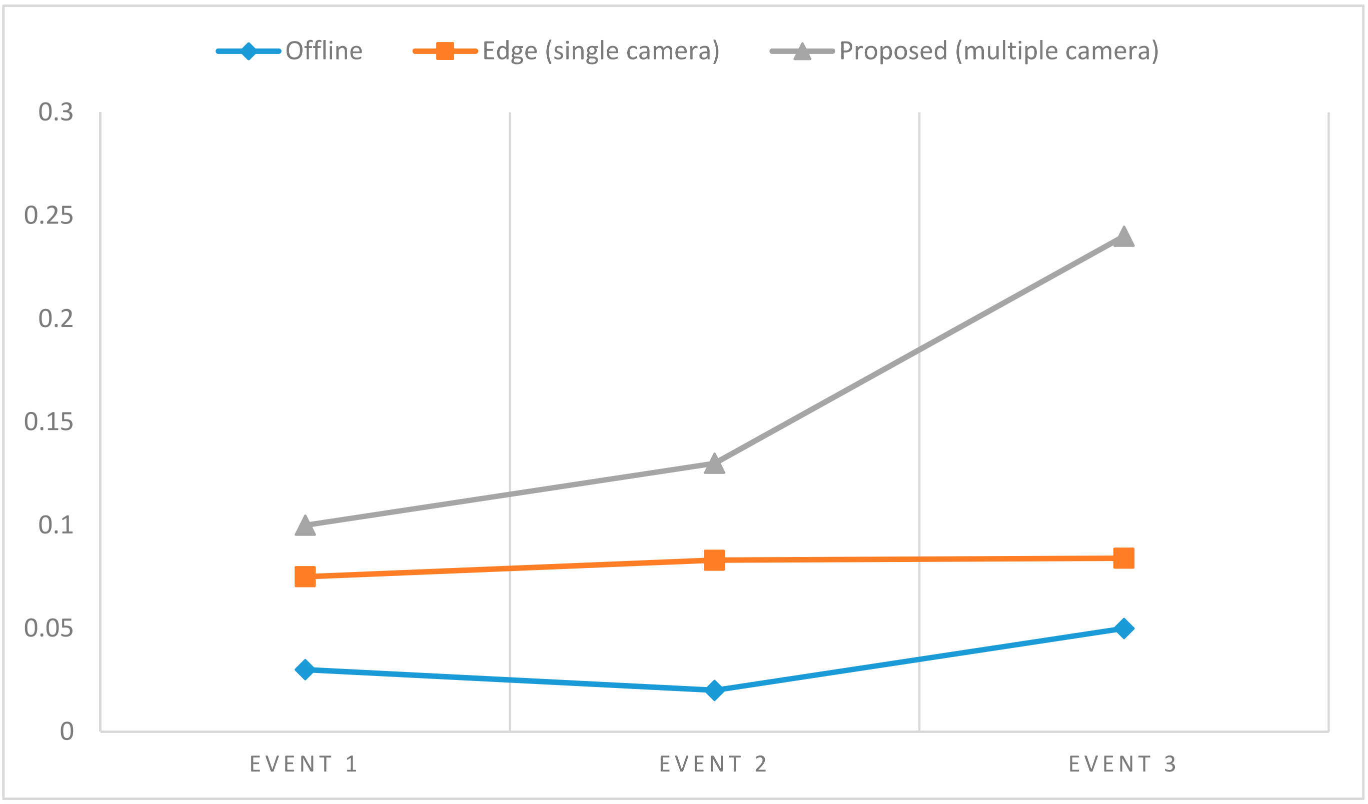


Figure 11. Relative latency values comparison of offline, edge, and the integrated architecture.

The latency values for Event 3 are the highest out of all of the scenarios because the complex event has more spatial and temporal relationships. The difference in latency for the integrated architecture is 0.85 times higher than that of the edge computing scenario, but this is not significant. For more complex events (i.e., complex events that include more spatial and temporal relationships), the developed architecture demonstrated significant latency growth in comparison to fewer complex events. Whilst this growth was not noticeable in the edge computing scenario, as the speed of offline processing was much higher than the other two scenarios, the latency difference between different events was relatively high in comparison to that of the edge computing architecture.

6. Discussion

Complex event processing (CEP) systems and complex event detection have been widely studied and applied in areas such as fraud detection [72], banking and insurance industries [73], flight diversion prediction [74], IoT [75], and vehicle information processing [76]. These systems can filter and aggregate events to provide semantic materials and constitute high-level meaningful events. Most of the existing CEP engines, such as knowledge-infused CEP [77], context-aware nested CEP [78], model4CEP [79], and intelligent machine to machine [80], do not present any solution for reducing or handling uncertainty in complex event detection.

One of the most interesting applications of CEP engines is their ability to detect complex events in video streams. Although there are several engines that can handle the uncertainties involved in complex events, there is still a gap in handling uncertainties in video CEP engines, such as VidCEP [20], Eventnet [81], and video event knowledge graph (VEKG) CEP [82]. Complex event detection in video streams involves various objects with different confidence levels, spatial relationships, and temporal relationships. Errors in each part affects the final event detection quality. Sources of uncertainty for CEP engines are either missing data or outliers [83]. Missing data are a major problem for CEP engines and increase the uncertainty of the final results.

The proposed integrated architecture leverages multiple camera streams and cloud computing to resolve the problem of missing data. Gathering information from different observation points allowed our system to resolve the problem of missing objects and simple events in camera streams. Based on the experimental results in Section 4, the accuracy of complex event detection increased in comparison to a traditional complex event detection system. As a result, it can be argued that the solution improved the accuracy of the complex event detector.

7. Conclusions

One of the most significant problems for complex event detection is the problem of missing objects and simple events in real-time camera streams. The missing object problem often occurs because of camera angles, low precision of object detection models, or low processing speeds and missing frames. This research addressed the second and third issues using high-speed YOLO computation with edge computing before utilising an integrated edge and cloud computing architecture that provided different viewpoints of the same scene using multiple cameras in order to increase the confidence level of object detection. This paper presents an IoSC based on an integrated edge and cloud computing architecture for complex event detection. For this, video stream results (i.e., detected objects and simple events) were obtained from several cameras directed at a common scene with different viewing angles. The designed architecture collected different parts of a complex event from multiple cameras and matched the event using the cloud services. By doing so, this architecture reduced the possibility of missing objects and simple events. Finally, we applied the OGC STA as an open-source international standard in order to improve the interoperability between smart cameras.

The experimental results of online event matching reveal the accuracy of both the object detection model and the speed effect final complex event. The object detection results demonstrate that the YOLO model delivered better accuracy for complex event processing as it missed fewer frames. In order to evaluate the designed architecture, F-scores of online event detection were used. They demonstrate that the accuracy of the developed integrated edge and cloud computing architecture was significantly greater than that of the edge computing architecture. The latency of the integrated architecture was 0.85 times more than that of the edge-based architecture. This developed architecture was more sensitive to the number of spatial and temporal relationships in relation to latency. Therefore, the differences between the developed architecture latency for edge computing and offline scenarios were more significant than for more complex events. Finally, an integrated cloud and edge computing architecture for IoSC was implemented to detect risk behaviours related to COVID-19 spread. The results demonstrate that the developed architecture had an average accuracy of 68%. This was higher than that of the single-camera edge computing scenario. In conclusion, multiple cameras delivered better results than single cameras in relation to complex event detection accuracy. The latency of the integrated architecture was greater than that of the edge-based computing architecture, but this difference did not significantly impact our case study of COVID-19 risk behaviour detection.

We intend to concentrate on increasing and verifying the accuracy of detecting simple and complex events from different sensor types for future research. For example, the detection of voices and images of objects from cameras and microphones enables us to more precisely detect complex events from different sources [58,84]. In addition, the next stage of our research will focus on complex event detection from cameras directed towards different places [85]. Moreover, a potential area of interest for further exploration would be developing a framework for building complex events from simple events derived from separate detection areas

Supplementary Materials: The following are available online at https://www.mdpi.com/2220-9964/10/2/81/s1.

Author Contributions: Research conceptualization, Sepehr Honarparvar, Sara Saeedi, Steve Liang, and Jeremy Squires; methodology and software, Sepehr Honarparvar and Jeremy Squires; validation, Sepehr Honarparvar and Sara Saeedi; writing—original draft preparation, Sepehr Honarparvar; writing—review and editing, Sepehr Honarparvar, Sara Saeedi, Steve Liang, and Jeremy Squires; Steve Liang supervised this research project. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The Developed Event Matching Pseudocode

Definition 1 (Event Vector). *An event stored in the event database is considered a vector with the following elements:*

O *= The array of detected object classes;*
Sr *= Spatial relationship vector which includes the spatial relationship expression and involved object classes;*
Tr *= Temporal relationship vector which includes the temporal relationship expression and involved spatial relationships;*
Rt *= An array (the root vector) of vectors including relationships between Tr. For example, an Rt looks like this:*
Rt = [R_1(Tr_1,Tr_2,relationship_expression1), . . . , Rn(Tr_{n-1},Tr_n,relationship_expression$_n$)]

Definition 2 (Detection Object Vector). *The detected object in every frame which includes the following elements:*

Bbox *= The rectangle of object bounding boxes is an array: [x_{min}, y_{min}, **width**, **height**];*
Class *= The class name of the detected object;*
Confidence *= The probability confidence value of the detected object.*
Trackid *= The ID assigned to every object when the tracking module is run.*

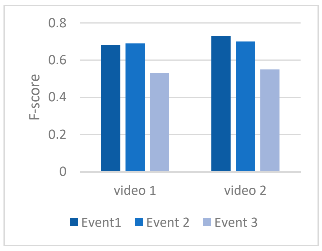
Pseudocode 1—Event Matching Process

```
begin
    c = captureFrame()
    spatial_rel=[]
    temporal_rel=[]
    while c
        t = captureTime()
        det_obj = objectDetector(c)
        if det_obj.siz>0 do
            for each event from eventPatterns do
                object_event = []
                for each obj_pattern from event.object do
                    if det_obj in obj_pattern do
                        object_event.add(det_obj)
                    end
                end
                if object_event.size() > 1 do
                    for each sr from event.Sr do
                        object1 = find(sr.object1.class,object_event)
```

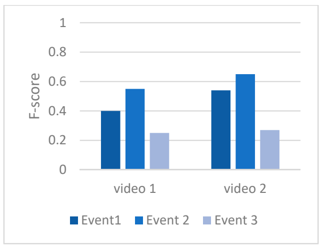
```
                                        object2 = find(sr.object2.class,object_event)
                                r = spatialRelation(object1,object2)
                                if r is True do
                                    s = (sr,obj1,obj2,t)
                                    spatial_rel.add(s)
                                end
                            end
                            for each temporal_rel from event.Tr do
                                sr1 = find(temporal_rel.s1,spatial_rel)
                                sr2 = find(temporal_rel.s2,spatial_rel)
                                event_detected = temporalRelation(sr1.t,sr2.t)
                                if event_detected is True do
                                    T = (sr,obj1,obj2,t)
                                    temporal_rel.add(T)
                                end
                            end
                            for each root_vector from event.Rt do
                                R1 = find(root_vector.T1,temporal_rel)
                                R2 = find(root_vector.T2,temporal_rel)
                                R = root_relationship(R1,R2)
                                if R is False do
                                    skipFrame()
                                end
                            end
                            return event
                        end
                    end
                end
            end
        end
```

Appendix B

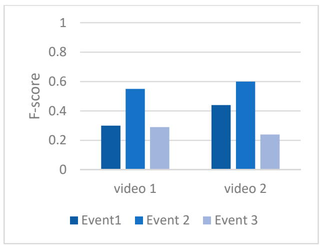
The detailed values for the matching accuracy of the integrated architecture and edge computing results for Camera1 and Camera2 using two videos from three events.



(a)



(b)



(c)

Figure A1. Event matching results. (**a**) The integrated designed architecture event detection results. (**b**) The single-camera edge-based architecture results for Camera1. (**c**) The single-camera edge-based architecture results for Camera2. Event 1 is coughing into hands and touching the door handle; Event 2 is coughing into arms and touching the door handle; Event 3 is coughing into hands, and/or arms, and touching the door handle.

References

1. Banerjee, S.; Wu, D.O. *Final Report from the NSF Workshop on Future Directions in Wireless Networking*; National Science Foundation: Washington, DC, USA, 2013.
2. Frankowski, G.; Jerzak, M.; Miłostan, M.; Nowak, T.; Pawłowski, M. Application of the Complex Event Processing system for anomaly detection and network monitoring. *Comput. Sci.* **2015**, *16*, 351–371.
3. Li, S.; Son, S.H.; Stankovic, J.A. Event detection services using data service middleware in distributed sensor networks. In *Information Processing in Sensor Networks*; Springer: Berlin/Heidelberg, Germany, 2003; pp. 502–517.
4. Fan, H.; Chang, X.; Cheng, D.; Yang, Y.; Xu, D.; Hauptmann, A.G. Complex event detection by identifying reliable shots from untrimmed videos. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 736–744.
5. Wu, E.; Diao, Y.; Rizvi, S. High-performance complex event processing over streams. In Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, Chicago, IL, USA, 27–29 June 2006; pp. 407–418.
6. Cugola, G.; Margara, A. Processing flows of information: From data stream to complex event processing. *ACM Comput. Surv. (CSUR)* **2012**, *44*, 1–62. [CrossRef]
7. Butakova, M.A.; Chernov, A.V.; Shevchuk, P.S.; Vereskun, V.D. Complex event processing for network anomaly detection in digital railway communication services. In Proceedings of the 2017 25th Telecommunication Forum (TELFOR), Belgrade, Serbia, 21–22 November 2017; pp. 1–4.
8. Terroso-Saenz, F.; Valdes-Vela, M.; Sotomayor-Martinez, C.; Toledo-Moreo, R.; Gomez-Skarmeta, A.F. A cooperative approach to traffic congestion detection with complex event processing and VANET. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 914–929. [CrossRef]
9. Mazon-Olivo, B.; Hernández-Rojas, D.; Maza-Salinas, J.; Pan, A. Rules engine and complex event processor in the context of internet of things for precision agriculture. *Comput. Electron. Agric.* **2018**, *154*, 347–360. [CrossRef]
10. Liu, X.; Cao, J.; Tang, S.; Guo, P. Fault tolerant complex event detection in WSNs: A case study in structural health monitoring. *IEEE Trans. Mob. Comput.* **2015**, *14*, 2502–2515. [CrossRef]
11. Terroso-Saenz, F.; Valdes-Vela, M.; Skarmeta-Gomez, A.F. A complex event processing approach to detect abnormal behaviours in the marine environment. *Inf. Syst. Front.* **2016**, *18*, 765–780. [CrossRef]
12. Jin, X.; Yuan, P.; Li, X.; Song, C.; Ge, S.; Zhao, G.; Chen, Y. Efficient privacy preserving Viola-Jones type object detection via random base image representation. In Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME), Hong Kong, China, 10–14 July 2017; pp. 673–678.
13. Chen, J.; Ran, X. Deep learning with edge computing: A review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
14. Li, J.Z.; Ozsu, M.T.; Szafron, D.; Oria, V. MOQL: A multimedia object query language. In Proceedings of the 3rd International Workshop on Multimedia Information Systems, Seoul, Korea, 22–26 October 2018; pp. 19–28.
15. Kuo, T.C.; Chen, A.L. Content-based query processing for video databases. *IEEE Trans. Multimedia* **2000**, *2*, 1–13. [CrossRef]
16. Aref, W.; Hammad, M.; Catlin, A.C.; Ilyas, I.; Ghanem, T.; Elmagarmid, A.; Marzouk, M. Video query processing in the VDBMS testbed for video database research. In Proceedings of the 1st ACM International Workshop on Multimedia Databases, New Orleans, LA, USA, 7 November 2003; pp. 25–32.
17. Lu, C.; Liu, M.; Wu, Z. Svql: A sql extended query language for video databases. *Int. J. Database Theor. Appl.* **2015**, *8*, 235–248. [CrossRef]
18. Kang, D.; Bailis, P.; Zaharia, M. BlazeIt: Optimizing declarative aggregation and limit queries for neural network-based video analytics. *arXiv* **2018**, arXiv:1805.01046. [CrossRef]
19. Jain, P.; Shukla, V.; Srinivasan, A.; de Castro Alves, A.; Hsiao, E. Support for a Parameterized Query/View in Complex Event Processing. U.S. Patent No. 8,713,049, 29 April 2014.
20. Yadav, P.; Curry, E. VidCEP: Complex Event Processing Framework to Detect Spatiotemporal Patterns in Video Streams. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 10–12 December 2019; pp. 2513–2522.
21. Medioni, G.; Cohen, I.; Brémond, F.; Hongeng, S.; Nevatia, R. Event detection and analysis from video streams. *IEEE Trans. Pattern Anal. Mach. Intell.* **2001**, *23*, 873–889. [CrossRef]
22. Li, Z.; Ge, T. History is a mirror to the future: Best-effort approximate complex event matching with insufficient resources. *Proc. VLDB Endow.* **2016**, *10*, 397–408. [CrossRef]
23. Ferreiros, J.; Pardo, J.M.; Hurtado, L.-F.; Segarra, E.; Ortega, A.; Lleida, E.; Torres, M.I.; Justo, R. ASLP-MULAN: Audio speech and language processing for multimedia analytics. *Proces. Leng. Nat.* **2016**, *57*, 147–150.
24. Yang, A.; Zhang, C.; Chen, Y.; Zhuansun, Y.; Liu, H. Security and privacy of smart home systems based on the Internet of Things and stereo matching algorithms. *IEEE Int. Things J.* **2019**, *7*, 2521–2530. [CrossRef]
25. Karthick, R.; Prabaharan, A.M.; Selvaprasanth, P. Internet of things based high security border surveillance strategy. *Asian J. Appl. Sci. Technol. (AJAST) Vol.* **2019**, *3*, 94–100.
26. Al-Sakran, H.O. Intelligent traffic information system based on integration of Internet of Things and Agent technology. *Int. J. Adv. Comput. Sci. Appl.* **2015**, *6*, 37–43.
27. Zhang, S.; Yu, H. Person re-identification by multi-camera networks for Internet of Things in smart cities. *IEEE Access* **2018**, *6*, 76111–76117. [CrossRef]
28. Sara Saeedi, J.L.; Liang, S.; Hawkins, B.; Chen, C.; Correas, I.; Starkov, I.; MacDonald, J.; Alzona, M.; Botts, M.; Mohammadi Jahromi, M.; et al. *OGC SCIRA Pilot Engineering Report*; Open Geospatial Consortium: Wayland, MA, USA, 2020.

29. García, C.G.; Meana-Llorián, D.; G-Bustelo, B.C.P.; Lovelle, J.M.C.; Garcia-Fernandez, N. Midgar: Detection of people through computer vision in the Internet of Things scenarios to improve the security in Smart Cities, Smart Towns, and Smart Homes. *Future Gener. Comput. Syst.* **2017**, *76*, 301–313. [CrossRef]
30. Chaudhry, R.; Ravichandran, A.; Hager, G.; Vidal, R. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 1932–1939.
31. Fan, H.; Luo, C.; Zeng, C.; Ferianc, M.; Que, Z.; Liu, S.; Niu, X.; Luk, W. F-E3D: FPGA-based acceleration of an efficient 3D convolutional neural network for human action recognition. In Proceedings of the 2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP), New York, NY, USA, 15–17 July 2019; pp. 1–8.
32. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
33. Han, Y.; Zhang, P.; Zhuo, T.; Huang, W.; Zhang, Y. Going deeper with two-stream ConvNets for action recognition in video surveillance. *Pattern Recogn. Lett.* **2018**, *107*, 83–90. [CrossRef]
34. Girshick, R. Fast r-cnn. In Proceedings of the 2015 IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
35. Ko, K.-E.; Sim, K.-B. Deep convolutional framework for abnormal behavior detection in a smart surveillance system. *Eng. Appl. Artif. Intell.* **2018**, *67*, 226–234. [CrossRef]
36. Gan, C.; Wang, N.; Yang, Y.; Yeung, D.-Y.; Hauptmann, A.G. Devnet: A deep event network for multimedia event detection and evidence recounting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 2568–2577.
37. Xiong, Y.; Zhu, K.; Lin, D.; Tang, X. Recognize complex events from static images by fusing deep channels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 1600–1609.
38. Jhuo, I.-H.; Lee, D. Video event detection via multi-modality deep learning. In Proceedings of the 2014 22nd International Conference on Pattern Recognition, Stockholm, Sweden, 24–28 August 2014; pp. 666–671.
39. Wu, Z.; Jiang, Y.-G.; Wang, X.; Ye, H.; Xue, X. Multi-stream multi-class fusion of deep networks for video classification. In Proceedings of the 24th ACM International Conference on Multimedia, Amsterdam, The Netherlands, 5–19 October 2016; pp. 791–800.
40. Habibian, A.; Mensink, T.; Snoek, C.G. Composite concept discovery for zero-shot video event detection. In Proceedings of the International Conference on Multimedia Retrieval, Glasgow, UK, 1–4 April 2014; pp. 17–24.
41. Mazloom, M.; Gavves, E.; van de Sande, K.; Snoek, C. Searching informative concept banks for video event detection. In Proceedings of the 3rd ACM conference on International conference on multimedia retrieval, Dallas, TX, USA, 16–19 April 2013; pp. 255–262.
42. Rastegari, M.; Diba, A.; Parikh, D.; Farhadi, A. Multi-attribute queries: To merge or not to merge? In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 3310–3317.
43. Dubba, K.S.; Cohn, A.G.; Hogg, D.C.; Bhatt, M.; Dylla, F. Learning relational event models from video. *J. Artif. Intell. Res.* **2015**, *53*, 41–90. [CrossRef]
44. Kang, D.; Emmons, J.; Abuzaid, F.; Bailis, P.; Zaharia, M. Noscope: Optimizing neural network queries over video at scale. *arXiv* **2017**, arXiv:1703.02529. [CrossRef]
45. Hsieh, K.; Ananthanarayanan, G.; Bodik, P.; Venkataraman, S.; Bahl, P.; Philipose, M.; Gibbons, P.B.; Mutlu, O. Focus: Querying large video datasets with low latency and low cost. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; pp. 269–286.
46. Choochotkaew, S.; Yamaguchi, H.; Higashino, T.; Shibuya, M.; Hasegawa, T. EdgeCEP: Fully-distributed complex event processing on IoT edges. In Proceedings of the 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), Ottawa, ON, Canada, 5 June 2017; pp. 121–129.
47. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 779–788.
48. Zhao, Z.-Q.; Zheng, P.; Xu, S.-t.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
49. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, San Juan, PR, USA, 17–19 June 1997; pp. 580–587.
50. Chen, H.; Wang, Y.; Wang, G.; Qiao, Y. Lstd: A low-shot transfer detector for object detection. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 5 February 2018.
51. Hou, X.; Wang, Y.; Chau, L.-P. Vehicle Tracking Using Deep SORT with Low Confidence Track Filtering. In Proceedings of the 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 18 September 2019; pp. 1–6.
52. Punn, N.S.; Sonbhadra, S.K.; Agarwal, S. Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLO v3 and Deepsort techniques. *arXiv* **2020**, arXiv:2005.01385.
53. Theagarajan, R.; Pala, F.; Zhang, X.; Bhanu, B. Soccer: Who has the ball? generating visual analytics and player statistics. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1749–1757.

54. Kurniawan, A.; Ramadlan, A.; Yuniarno, E. Speed Monitoring for Multiple Vehicle Using Closed Circuit Television (CCTV) Camera. In Proceedings of the 2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM), Surabaya, Indonesia, 26 November 2018; pp. 88–93.
55. Chmielewska, A.; Marianna, P.; Marciniak, T.; Dabrowski, A.; Walkowiak, P. Application of the projective geometry in the density mapping based on CCTV monitoring. In Proceedings of the 2015 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, Poland, 3 May 2015; pp. 179–184.
56. Gonzalez-Sosa, E.; Vera-Rodriguez, R.; Fierrez, J.; Tome, P.; Ortega-Garcia, J. Pose variability compensation using projective transformation for forensic face recognition. In Proceedings of the 2015 International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, Germany, 9 September 2015; pp. 1–5.
57. Miller, E.; Banerjee, N.; Zhu, T. Smart Homes that Detect Sneeze, Cough, and Face Touching. *Smart Health* **2020**, *19*, 100170. [CrossRef]
58. Liang, S.H.; Saeedi, S.; Ojagh, S.; Honarparvar, S.; Kiaei, S.; Jahromi, M.M.; Squires, J. An Interoperable Architecture for the Internet of COVID-19 Things (IoCT) Using Open Geospatial Standards—Case Study: Workplace Reopening. *Sensors* **2021**, *21*, 50. [CrossRef]
59. Maeda, K. Performance evaluation of object serialization libraries in XML, JSON and binary formats. In Proceedings of the 2012 Second International Conference on Digital Information and Communication Technology and It's Applications (DICTAP), Bankok, Thailand, 16 May 2018; pp. 177–182.
60. Yaghmazadeh, N.; Wang, X.; Dillig, I. Automated migration of hierarchical data to relational tables using programming-by-example. *Proc. VLDB Endow.* **2018**, *11*, 580–593. [CrossRef]
61. Chasseur, C.; Li, Y.; Patel, J.M. Enabling JSON Document Stores in Relational Systems. In Proceedings of the WebDB, New York, NY, USA, 23 June 2013; pp. 14–15.
62. Kotsev, A.; Schleidt, K.; Liang, S.; Van der Schaaf, H.; Khalafbeigi, T.; Grellet, S.; Lutz, M.; Jirka, S.; Beaufils, M. Extending INSPIRE to the Internet of Things through SensorThings API. *Geosciences* **2018**, *8*, 221. [CrossRef]
63. Liang, S.; Huang, C.-Y.; Khalafbeigi, T. *OGC SensorThings API Part 1: Sensing*; Version 1.0; Open Geospatial Consortium: Wayland, MA, USA, 2016.
64. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Hagen, S.; Schaub, T. *The Geojson Format*; Internet Engineering Task Force (IETF): Wilmington, NC, USA, 2016.
65. Horbiński, T.; Cybulski, P. Similarities of global web mapping services functionality in the context of responsive web design. *Geod. Cartogr.* **2018**, *67*, 159–177. [CrossRef]
66. Horbiński, T.; Lorek, D. The use of Leaflet and GeoJSON files for creating the interactive web map of the preindustrial state of the natural environment. *J. Spat. Sci.* **2020**. [CrossRef]
67. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Glasgow, UK, 23–28 August 2020; pp. 740–755.
68. Mittal, A.; Zisserman, A.; Torr, P.H. Hand detection using multiple proposals. In Proceedings of the BMVC, Claverton, UK, 8 September 2020; p. 5.
69. Bashiri, F.S.; LaRose, E.; Peissig, P.; Tafti, A.P. MCIndoor20000: A fully-labeled image dataset to advance indoor objects detection. *Data Brief* **2018**, *17*, 71–75. [CrossRef] [PubMed]
70. Kuznetsova, A.; Rom, H.; Alldrin, N.; Uijlings, J.; Krasin, I.; Pont-Tuset, J.; Kamali, S.; Popov, S.; Malloci, M.; Kolesnikov, A. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv* **2018**, arXiv:1811.00982. [CrossRef]
71. Abdulla, W. Mask R-CNN for Object Detection and Instance Segmentation on Keras and Tensorflow. 2017. Available online: https://github.com/matterport/Mask_RCNN (accessed on 18 December 2020).
72. Correia, I.; Fournier, F.; Skarbovsky, I. The uncertain case of credit card fraud detection. In Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems, Barcelona, Spain, 3–7 July 1995; pp. 181–192.
73. Adi, A.; Botzer, D.; Nechushtai, G.; Sharon, G. Complex event processing for financial services. In Proceedings of the 2006 IEEE Services Computing Workshops, Chicago, IL, USA, 18–22 September 2006; pp. 7–12.
74. Cabanillas Macías, C.; Curik, A.; Di Ciccio, C.; Gutjahr, M.; Mendling, J.; Prescher, J.; Simecka, J. Combining Event Processing and Support Vector Machines for Automated Flight Diversion Predictions. In Proceedings of the 1st International Workshop on Modeling Inter-Organizational Processes and 1st International Workshop on Event Modeling and Processing in Business Process Management co-located with Modellierung, Vienna, Austria, 19 March 2014; pp. 45–47.
75. Chen, C.Y.; Fu, J.H.; Sung, T.; Wang, P.-F.; Jou, E.; Feng, M.-W. Complex event processing for the internet of things and its applications. In Proceedings of the 2014 IEEE International Conference on Automation Science and Engineering (CASE), Taipei, Taiwan, 19 August 2014; pp. 1144–1149.
76. Nielsen, S.; Chambers, C.; Farr, J. Systems and Methods for Complex Event Processing of Vehicle Information and Image Information Relating to a Vehicle. U.S. Patent No. 8,560,164, 15 October 2013.
77. Bonino, D.; De Russis, L. Complex event processing for city officers: A filter and pipe visual approach. *IEEE Int. Things J.* **2017**, *5*, 775–783. [CrossRef]
78. Peng, S.; He, J. Efficient Context-Aware Nested Complex Event Processing over RFID Streams. In Proceedings of the International Conference on Web-Age Information Management, Nanchang, China, 3 June 2016; pp. 125–136.
79. Djedouboum, A.C.; Ari, A.; Adamou, A.; Gueroui, A.M.; Mohamadou, A.; Aliouat, Z. Big data collection in large-scale wireless sensor networks. *Sensors* **2018**, *18*, 4474. [CrossRef] [PubMed]

80. Bruns, R.; Dunkel, J.; Masbruch, H.; Stipkovic, S. Intelligent M2M: Complex event processing for machine-to-machine communication. *Expert Syst. Appl.* **2015**, *42*, 1235–1246. [CrossRef]
81. Ye, G.; Li, Y.; Xu, H.; Liu, D.; Chang, S.-F. Eventnet: A large scale structured concept library for complex event detection in video. In Proceedings of the 23rd ACM international conference on Multimedia, Brisbane, Australia, 26–30 October 2015; pp. 471–480.
82. Yadav, P.; Curry, E. VEKG: Video Event Knowledge Graph to Represent Video Streams for Complex Event Pattern Matching. In Proceedings of the 2019 First International Conference on Graph Computing (GC), Laguna Hills, CA, USA, 25–27 September 2019; pp. 13–20.
83. Tawsif, K.; Hossen, J.; Raja, J.E.; Jesmeen, M.; Arif, E. A Review on Complex Event Processing Systems for Big Data. In Proceedings of the 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP), Kota Kinabalu, Indonesia, 26 March 2018; pp. 1–6.
84. Saeedi, S.; Moussa, A.; El-Sheimy, N. Context-Aware Personal Navigation Using Embedded Sensor Fusion in Smartphones. *Sensors* **2014**, *14*, 5742–5767. [CrossRef] [PubMed]
85. Ojagh, S.; Saeedi, S.; Liang, S.H.L. A Person-to-Person and Person-to-Place COVID-19 Contact Tracing System Based on OGC IndoorGML. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 2.