INTELLIGENT REAL-TIME VIDEO SURVEILLANCE SYSTEM USING 2-D SAD MOTION DETECTION AND MASCOT METHODOLOGY FOR MODULAR EMBEDDED DESIGN

Eze Ukamaka. J.[1], Bakare Kazeem[2], Ndubuisi Paul-Darlington Ibemezie[3],
Madonna University, Nigeria[1], Oska.Jo and Partners Ltd[2], Federal Polytechnic Ngodo-Isuochi, Abia State, Nigeria[3]
ukamakajoe@gmail.com[1], bakarekazeem1000@gmail.com[2], ndudarla@gmail.com[3]

ABSTRACT

This paper presents an intelligent real-time video surveillance system using a 2-D Sum of Absolute Differences (SAD) motion detection algorithm, integrated with the MASCOT (Modular Approach to Software Construction Operation and Test) methodology to ensure scalability, modularity, and reliability. The system processes live video fed from multiple IP/Bluetooth cameras, detects motion using SAD values exceeding a defined threshold, and transmits and stores frames over a TCP/IP network. By applying MASCOT, we decompose the system into well-defined activity and communication modules, enabling formal verification, system-level testing, and easier maintainability. Simulation results on a DSP platform (TMS320C6416) demonstrated the feasibility of the system for real-time motion detection and multi-camera synchronization, with potential applications in security monitoring, smart environments, and automated surveillance.

Keywords/Phrases: Video Surveillance, Motion Detection, Real-Time System, DSP, Modular Design, Multi-camera Synchronization

1. INTRODUCTION

Video surveillance systems have evolved rapidly, with the demand for real-time intelligent monitoring growing across multiple domains. Camera-based surveillance is an important technology used to monitor people, assets and places, for the applications of increasing physical security (L. A. Hassnawi, *et al.,* 2022). Multi-camera video surveillance systems have generated fast growing interest in recent time, because systems relying on a single video camera tend to restrict both the visual coverage and the total resolution available, which usually impose undesirable constraints (R. Rosales *et al.,* 2020). The technological evolution of vision surveillance systems starts with video-based surveillance systems consisting of analogue Closed-Circuit Television (CCTV) systems, i.e. a number of systems with potential applications in control and image processing suggested in Maria Valera (2016). Traditional systems often suffer from scalability and maintenance issues. This research proposes a novel integration of a motion-based surveillance algorithm with the MASCOT methodology to enhance system design, testing, and extensibility. Prior research in motion detection has widely utilized SAD, background subtraction, and optical flow. However, few efforts combine low-level video processing with high-level architectural methodologies like MASCOT. This integration addresses the challenge of constructing real-time, embedded, and modular surveillance systems. MASCOT (Modular Approach to Software Construction Operation and Test) is an extension of Real Time Networks (E. J. Ong, 2019). MASCOT is a software design method for the design and implementation of large real-time concurrent systems. Ken Jackson and Hugo Simpson originated its essential concepts at the UK Royal Signals and Radar Establishment (RSRE) during the period 1971-5 [Bate 1986]. The first technical notes (MASCOT 1) were published in 1978 [IECCA and MUF 1978a]. The first official handbook was issued in 1983 (MASCOT-2) (IECCA and MUF, 1983b). Further refinements and extensions were continued until the official standard for MASCOT-3 (IECCA and MUF, 1987c) was published in 1987. The RTN approach in MASCOT/ DORIS differs from most other design methods, because design solutions are expressed in terms of a set of concurrent components, which work independently and interact. An IDA component allows several independently scheduled single sequential program threads (activities) to be simultaneously active or temporarily suspended. The IDA safeguards the integrity of data by using the minimum amount of mutual exclusion needed to avoid critical data clashes. The IDA maintains the propagation of data in the network by providing cross stimulation between activities. MASCOT takes requirement specifications obtained by other means as its starting point. MASCOT data flow networks should be static. Activities should not be created dynamically and the system network should remain invariant at run time to avoid hazards in terms of unconstrained resource demands and nondeterministic timing.

However, it seems that special measures can be used for those applications that cannot be implemented without dynamic network creation. Even though these measures are not published, one of them (taken from a private conversation [H. Simpson2005]) may consist in creating and building a new component, and then it is inserted into the network. Once it is established, the old component is removed from the network. Another measure may be the use of a protocol, called Remote Thread Invocation (RTI), which activates a thread at run-time. MASCOT assumes that the software system is being designed for a particular virtual machine called the MASCOT kernel, and the implementation of this virtual machine on a specific computer or a set of computers (depending on the configuration) is a separate problem. The dominant RTN principle embodied in MASCOT is that the flow of data through the system is controlled solely by a set of concurrent software process (E. J. Ong, 2019), which means that MASCOT uses the concept of data flow network between concurrent processes that constitute the network, as the means for expressing software structure.

II. MATERIALS AND METHOD

2.1 Material

2.1.1 Hardware & Software Requirements

The hardware and software requirements are TMS320C6416 DSP board, D.SignT Ethernet daughter card, Simulink with Video/Image Processing and Target Support Packages, TCP/IP or Bluetooth communication protocol and communication protocol

2.2 Methods

2.2.1 2-D Motion Detection Algorithm

The 2-D SAD block computes the absolute difference between pixel intensities of successive frames. When the SAD value exceeds a preset threshold, motion is flagged, and the relevant frame is stored and displayed.

2.2.2 System Design Using MASCOT

To structure the system, the following MASCOT components were defined as activity and communication modules. Activity Modules are video capture module, frame send module, motion etc. While communication modules are raw frame channel, transmit frame channel, SAD etc. Each module performs a specific task and communicates with others using clearly defined data channels, allowing parallel development and testing.

2.3 Simulation and Experimental work

This work used motion detection method, quality-of-view (QOV) and region of interest (ROI) configurations to achieve its aims and objectives. It also used MATLAB to develop a program using (Simulik), this program recognizes, tracks moving objects as they move through the camera's field of view and save the live video inputs for reference purpose. MATLAB was also used to develop a program that can filter the image captured in other to present us with a clear and informative data.

2.3.1 General Motion Equation and Simplification

The relative location of the camera, represented by the vector η to the feature point can be determined from equation 1.

$$\eta = R_{cb}R_{be}(\xi - r) - R_{cb}\delta \quad \text{........ (1)}$$

Where Rcb is a coordinate transform matrix from body-fixed to camera-fixed coordinates. These coordinate systems are related by the angles $[\phi_c, \theta_c, \psi_c]$. Similarly, Rbe represents the coordinate transform matrix from earth-fixed to body-fixed coordinate coordinates, related by the angles $[\phi_a, \theta_a, \psi_a]$. ξ and r here represent the location of the feature point and the object, respectively, in the earth-fixed coordinate system. Finally, δ is a vector describing the relative location of the camera to the center of gravity of the object in the body-fixed coordinate system. Thus, equation1 describes generally any location and attitude object and camera. Since simulating the vibration of a mounted camera is also a concern of this research work, a number of simplifications were made to this equation. Firstly, the camera is located at a height of 9 feet tall, that is equivalent to 274.5cmbut for the purpose of this work 270cm was used as the located height of the camera.

2.3.2 Image data equations and simplification

Once the relative position of the camera to the feature point has been determined, equations 2 and 3 described the location in the x and y directions on the plane of the image. These coordinates are represented by the variables ν and μ, respectively. The variable f describes the focal length of the camera, and the vector c accounts for any offset of the camera's center from its coordinates as described earlier by the variable δ.

$$\nu = f\frac{\eta_2 - c_2}{\eta_3 - c_3} \quad \text{........... (2)}$$

$$\mu = f\frac{\eta_1 - c_1}{\eta_3 - c_3} \quad \text{.............. (3)}$$

Equations 3.2 and 3.3 are defined for any point other than η3 equal to zero, but clearly an actual camera would be able to record points consistent with its field of view. Therefore, Causey et al. define a maximum and minimum value for ν and μ using γh and γv, called the horizontal and vertical fields of view, respectively. These in turn define the maxima and minima for the image coordinates using the relationship in equations 4 and 5

$$[\underline{\nu}, \bar{\nu}] = [-f\tan(\gamma_h), f\tan(\gamma_h)] \quad \text{.... (4)}$$

$$[\underline{\mu}, \bar{\mu}] = [-f\tan(\gamma_v), f\tan(\gamma_v)] \text{... (5)}$$

However, these conditions do not completely eliminate all points which should not be included. This is because equations (2) and (3) do not exclude points which happen to be behind the camera, which would be included on the opposite side. Once all of equations were simplified and appropriate assumptions made, certain parameters still need to be defined. These values are shown in Table 1 below, after measurements.

Table 1: Specified parameters

Focal length	50 Hz
Horizontal field of view	330cm
Vertical field of view	270cm
x-velocity	165.00m/s
y-velocity	135.00m/s
z-velocity	30m/s
x-initial position	0
y-initial position	0
z-initial position	0
Camera roll angle	450
Camera pitch angle	350
Camera yaw angle	250

Feature points, in order to complete the definition of the simulation parameters, a set of feature points is required. Therefore, a vector ξ was defined containing the outside surface of a building. Table 2 describes the location and dimensions of the building.

Table 2: Building Specifications

Width (x-direction)	350cm
Length (y-direction)	270cm
Height (z-direction)	150
Bottom-left ground coordinate	0,0

2.3.3 Automatic Camera Selection

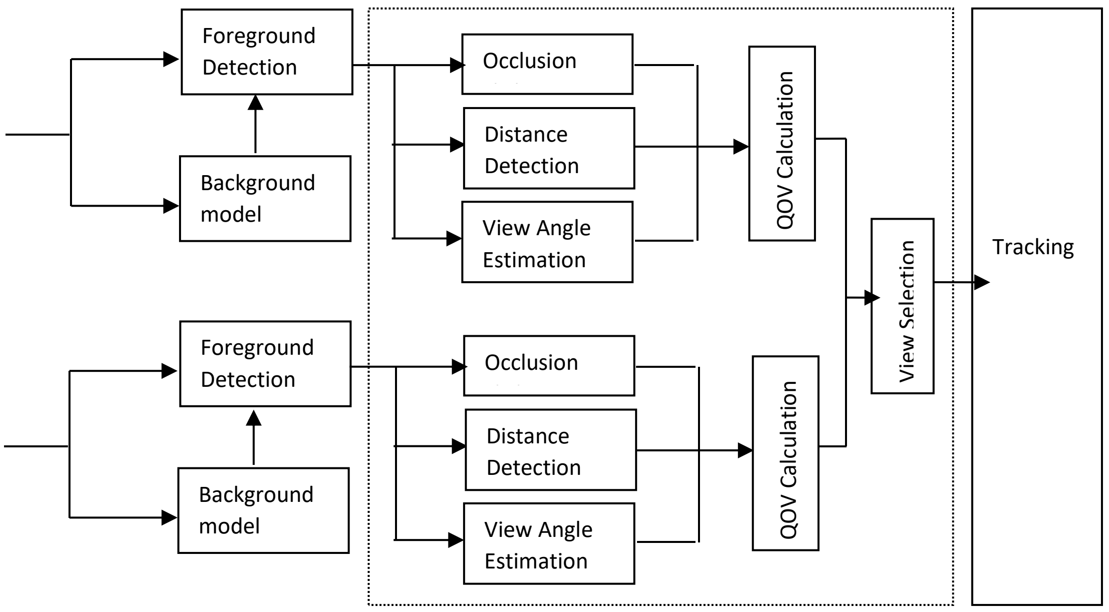

Figure 1: Surveillance camera System Architecture

Automatic camera selection is processed based on QOV calculations, which includes three primitive sub-tasks (occlusioncheck, distance detection, and view angle estimation). Finally, tracking is applied on the most informative camera. In the following sections, we describe the details about each subtask and how QOV is measured.

Quality-Of-View (QOV) Definition- In most applications, considering video data from all cameras is unnecessary because video data from some cameras are less informative and sometimes even misleading. Figure 2 shows the same subject's images captured from various view directions. Some images in the bottom row is captured from the subject's front view, and is more informative compared to other images for many applications, such as face tracking. The corresponding cameras of images in the top row obviously perform poorly in this case. This uninformative video data may trigger wrong alarm, and should not be considered in the system.

2.3.4 A 2-D SAD Motion Detection

The program uses the 2-D SAD block to detect motion in the video sequence. When the sum of absolute differences (SAD) value of a particular frame exceeds a threshold, the program records this video frame and displays it in the Motion Frames window.

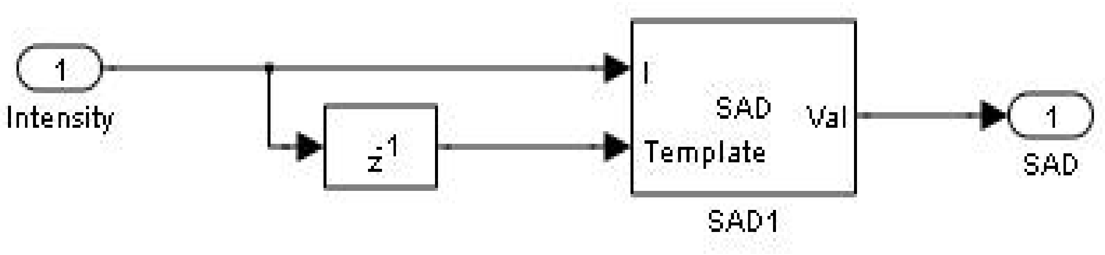

Figure 2: A 2-D SAD (Sum of Absolute Difference) Block Diagram

2.3.5 Mascot- RTN Principles

The MASCOT (Modular Approach to Software Construction Operation and Test) methodology provides a formal and structured approach to software development, especially for real-time systems. It emphasizes the early definition and partitioning of system components, promoting both temporal and spatial decoupling to ensure predictable performance. Every step from design to implementation is clearly defined, promoting disciplined development practices. It encourages a modular structure where functional and constructional elements align closely, aiding integration and maintenance. Supports the development of systems with reliable and deterministic temporal behavior. Facilitates incremental testing—from individual modules to grouped components—enhancing reliability. Utilizes a small, easily implemented runtime executive. Applicable across all stages of the software lifecycle, with support for system evolution and flexibility in accommodating design changes.

MASCOT emphasizes the construction of systems as real-time networks, rather than monolithic programs. It provides both a textual design language and a graphical network diagram to represent and manage system architecture dynamically. Enforces strong partitioning to isolate safety-critical functions, minimizing the risk of interference or corruption. Allows designers to adapt to changing requirements with minimal disruption. Promotes the use of templates for component reuse across projects. Its design visibility and controls enhance project management and maintainability by offering clear visibility into the design process and evolving system structure. Supports the identification and definition of independent execution units (activities) early in development, aiding in the distributed system representation.

2.4 The MASCOT Network Design

The Real-Time Network (RTN) approach of MASCOT tackles system complexity by leveraging concurrency as a core design strategy. A defining feature of this approach is the explicit separation of components into two main types: namely Activity Components (Active Elements) An Activity represents a fundamental processing unit in a MASCOT network. It is a single sequential thread of execution that can be scheduled independently, allowing all activities to execute concurrently in principle. These activities perform computation and interact with each other through shared data components. Intercommunication Data Areas (IDAs) (Passive Elements), An IDA serves as either a shared data store or a medium for information transmission. It is an encapsulated data structure, with its internal representation hidden from the components that use it.

In MASCOT, both activities and IDAs are defined using templates during the design phase: A template acts similarly to a class in object-oriented programming (OOP)—though with a different conceptual foundation. A component is created by instantiating a template, allowing for reuse and modularity across different system configurations. Each template includes: Interface Specifications – Define how components connect and communicate with one another. Definitions Section – Includes data types, constants, and interface details required by the component. Name Section – Assigns a unique identifier and indicates its classification (not to be confused with OOP "class"). These templates, once defined, are stored in a MASCOT database that manages the design's textual representations. The structured nature of templates ensures that the same component can be reused across different system designs or execution environments.

Table 3: Activity Modules (Tasks or Processes)

S/N	**Module Name**	**Description**
1	VideoCaptureModule	Captures live video from a camera (IP/Bluetooth)
2	FrameSendModule	Sends video frames to DSP via TCP/IP
3	MotionDetectionModule	Computes 2-D SAD values between frames
4	ThresholdComparisonModule	Compares SAD values with threshold
5	FrameRecorderModule	Saves frames where motion is detected
6	CameraSyncModule	Manages synchronization between cameras
7	ReconstructionModule	Performs 3-D reconstruction of scenes
8	ControlInterfaceModule	Host-side GUI/control (Simulink)
9	ParameterAdjustmentModule	Allows dynamic threshold change

Table 4: Communication Modules (Data Channels)

Channel Name	From Module	To Module	Description

RawFrameChannel	VideoCaptureModule	FrameSendModule	Raw video frames
TransmitFrameChannel	FrameSendModule	MotionDetectionModule	Frames sent to DSP
SADDataChannel	MotionDetectionModule	ThresholdComparisonModule	SAD values
ValidFrameChannel	ThresholdComparisonModule	FrameRecorderModule	Motion frames
SyncSignalChannel	CameraSyncModule	All Camera Modules	Vertical sync signal
3DDataChannel	FrameRecorderModule	ReconstructionModule	Captured motion frames
ControlChannel	ControlInterfaceModule	All modules	Start/Stop, threshold, view angle, etc.

2.5 MASCOT System Diagram

These are control signals used to trigger or manage the timing and behavior of modules (not to be confused with data flow): Control signals from the ControlInterfaceModule to: Start/stop surveillance, adjust motion threshold, select master sync camera, sync pulse from the CameraSyncModule to all camera nodes and trigger from ThresholdComparisonModule to FrameRecorderModule to record a frame.

The design

```
scss
CopyEdit
[VideoCaptureModule] --> (RawFrameChannel) --> [FrameSendModule]
  --> (TransmitFrameChannel) --> [MotionDetectionModule]
    --> (SADDataChannel) --> [ThresholdComparisonModule]
      --> (ValidFrameChannel) --> [FrameRecorderModule]
        --> (3DDataChannel) --> [ReconstructionModule]

[CameraSyncModule] --> (SyncSignalChannel) --> [VideoCaptureModule] xN

[ControlInterfaceModule] --> (ControlChannel) --> All modules
```

2.6 Mapping of the SAD-Based Surveillance into MASCOT

Mapping current SAD-based surveillance system into the MASCOT (Modular Approach to Software Construction Operation and Test) methodology, requires translate the existing video surveillance system using 2-D SAD and TCP/IP into a modular, MASCOT-compliant architecture with clearly defined activity modules, communication modules, and control interfaces. The system performs the following core functions: video capture from camera (IP/Bluetooth), frame transmission to DSP via TCP/IP, motion detection using 2-D SAD, motion threshold comparison, frame capture (Storage), synchronization of multiple cameras, 3-D reconstruction from multiple views and control & monitoring (host interface).

Table 5: MASCOT Mapping

Benefit	Impact on Your System
Modularity	Each functional block can be developed and tested independently.
Testability	You can simulate each module in isolation before system integration.
Scalability	Easy to add more camera modules or processing pipelines.
Maintainability	Clear separation of processing logic and communication makes debugging simpler.
Real-Time Scheduling	MASCOT supports formal timing analysis for real-time guarantees.

Table 6: Comparison of SAD-Based Motion Detection and MASCOT Methodology

Comparison of criterion	SAD-Based Motion Detection	MASCOT Methodology
Primary Focus	Enabled by deployment on DSP (e.g., TMS320C6416) with Simulink	Software architecture design for real-time and embedded systems
Core Principle	Uses TCP/IP or Bluetooth for host-target frame transmission	Promotes modular and decoupled task-based system construction
Application Domain	Primarily focused on motion logic and camera synchronization	General-purpose for control, signal processing, avionics, and telemetry systems
System Type	Limited—tailored for specific surveillance use case	Design methodology for real-time embedded systems
Real-Time Capability	Not inherently emphasized	Supports strict real-time requirements via scheduling and communication design
Communication Mechanism	Easy to integrate in Matlab/Simulink + DSP toolchain	Employs Data Streams and Control Signals between tasks/modules
Modularity	Fast, hardware-deployable, efficient in bandwidth and memory usage	Highly modular; uses Activity Modules and Communication Modules
Reusability & Scalability	Limited software-level abstraction, not a general software methodology	High—designed for reuse across many real-time systems
Testing and Maintenance Support	Enabled by deployment on DSP (e.g., TMS320C6416) with Simulink	Built-in testability and maintainability through defined module interfaces
Ease of Integration	Uses TCP/IP or Bluetooth for host-target frame transmission	Requires a formal design process and specific development tools
Strengths	Primarily focused on motion logic	Systematic design, great for

	and camera synchronization	safety-critical systems
Weaknesses	Limited—tailored for specific surveillance use case	Not optimized for video/image processing or multimedia data directly
Testing and Maintenance Support	Not inherently emphasized	
	Easy to integrate in Matlab/Simulink + DSP toolchain	

The differences are - the SAD-based system is a domain-specific application that solves a practical problem in motion detection and surveillance, while MASCOT is a design philosophy and methodology for engineering large-scale, reliable real-time systems. MASCOT provides a blueprint for designing the architecture of systems like the SAD-based motion detector but doesn't define how motion detection should be done. The SAD-based method implements a specific function within a system that could be built following MASCOT principles. SAD-based surveillance system could be redesigned or restructured using MASCOT, which would help improve modularity, testability, and maintainability, especially if scaled up to a city-wide or enterprise-level surveillance platform.

III. RESULTS

3.1 Surveillance Recording Results

The Motion Threshold window displays the threshold value in magenta, and plots the SAD values for each frame in yellow. Any time the SAD value exceeds the threshold, the model records the video frame. The Motion frames window shows the last recorded video frame. In this window, the Source frame value steadily increases as the video runs and the Captured frame value indicates the total number of frames recorded by the model.

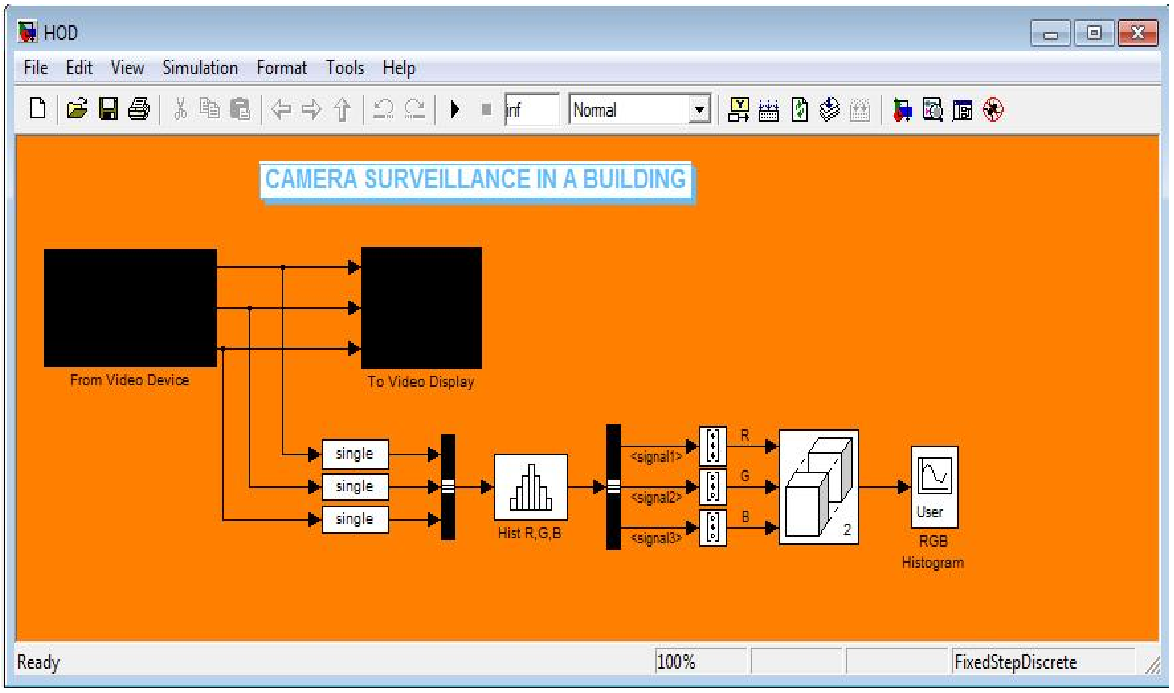

Figure 3: Surveillance Simulation

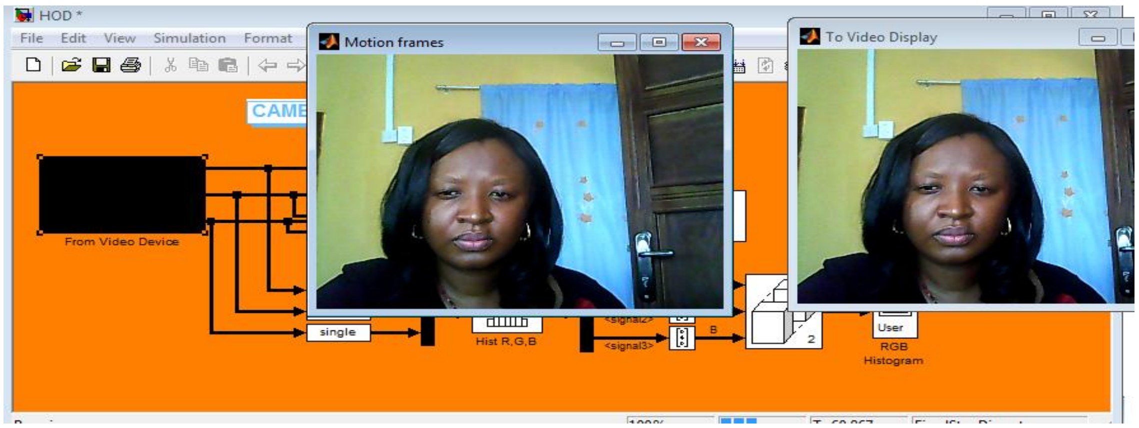

Figure 4: Display Original and motion frame

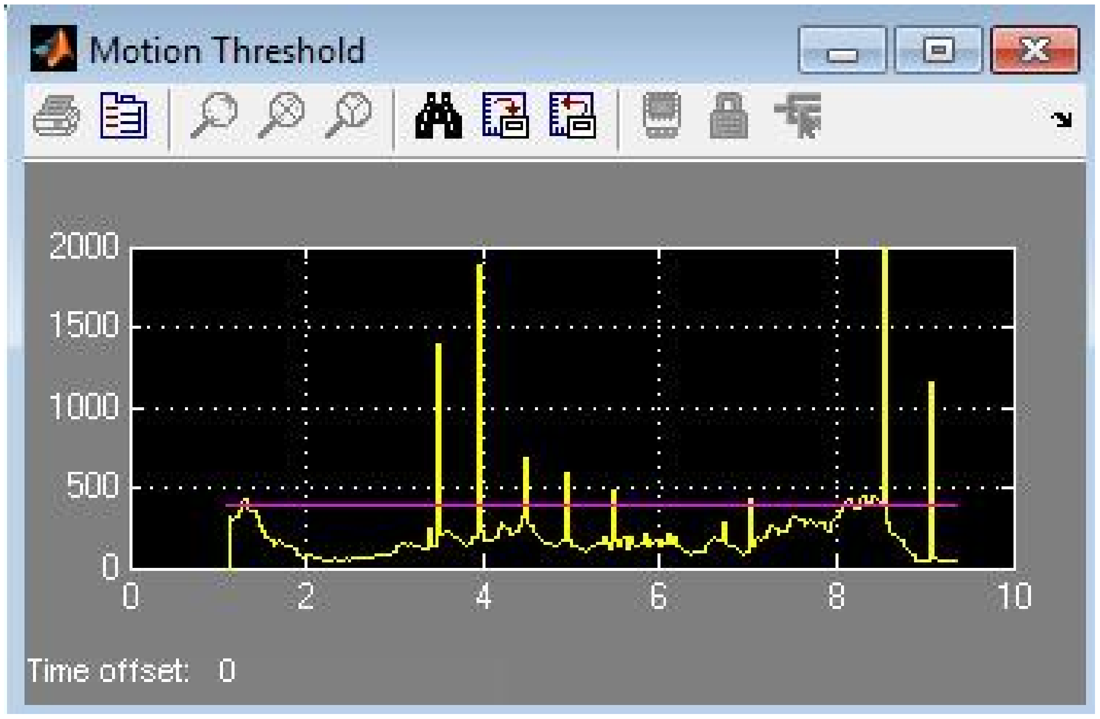

Figure 5: Motion Threshold

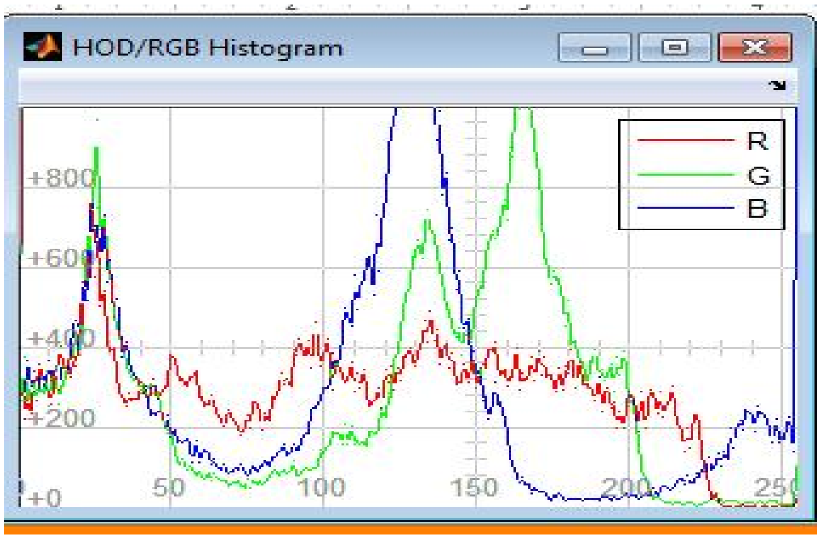

Figure 6: Histogram display Video Surveillance over TCP/IP Network

Instrument Control Toolbox provides Simulink blocks for sending and receiving data over TCP/IP and UDP networks. This program uses the TCP/IP or Bluetooth to Send and receive blocks to perform Video Surveillance. The Simulink model demonstrates a video surveillance recording on your Texas Instruments DSP platform using Target Support Package TC6. The motion detection algorithm is implemented in Simulink and deployed to a TIC6000 signal processor. This program requires Simulink, Video and Image Processing Blockset, and Target Support Package for TC6 to open the model.The program also requires the following hardware TMS320C6416 DSK / EVM board, D.signT DSK-91C111 Ethernet daughter card for C6416 DSK target and Ethernet cable.

IV. ANALYSIS

While the generated code is running on the target, a host side Simulink model simultaneously sends video frames to the target via TCP/IP protocol. The target receives video frames sent by the host side Simulink model, computes the sum of the absolute value of differences (SAD) between successive video frames, and returns an estimate of motion. When the motion estimate value exceeds a threshold, the target increments a counter and sends the corresponding frame back to the host using the TCP/IP block. You can also adjust the motion threshold using the host side Simulink model.

4.1 Running the Program

Open the target model and double click "Build Reload & Run" to build, load and run the DSP code. Once the code is generated, it will bring up the host side model. Run the host side model to watch the video surveillance algorithm using motion detection.

Using a system that incorporate multiple visible and Bluetooth BT camera, the vertical sync signal from one BT Bluetooth was used as a master to synchronization was performed by a custom auxiliary board embedded into the cabinet of the final system. Motion analysis of the image data collected could then be used to map data from the three asynchronously free IP/Bluetooth Camera unto the 3-D reconstructed images. In operation, 1Gbyte DDR memory in each of the embedded vision systems is capable of storing up to 155 of images from each of the camera. As image are captured, Bayer interpolation from the visible light cameras is performed and transferred to each of the computer's hard drives. 3_D images construction can then be performed using the visible data

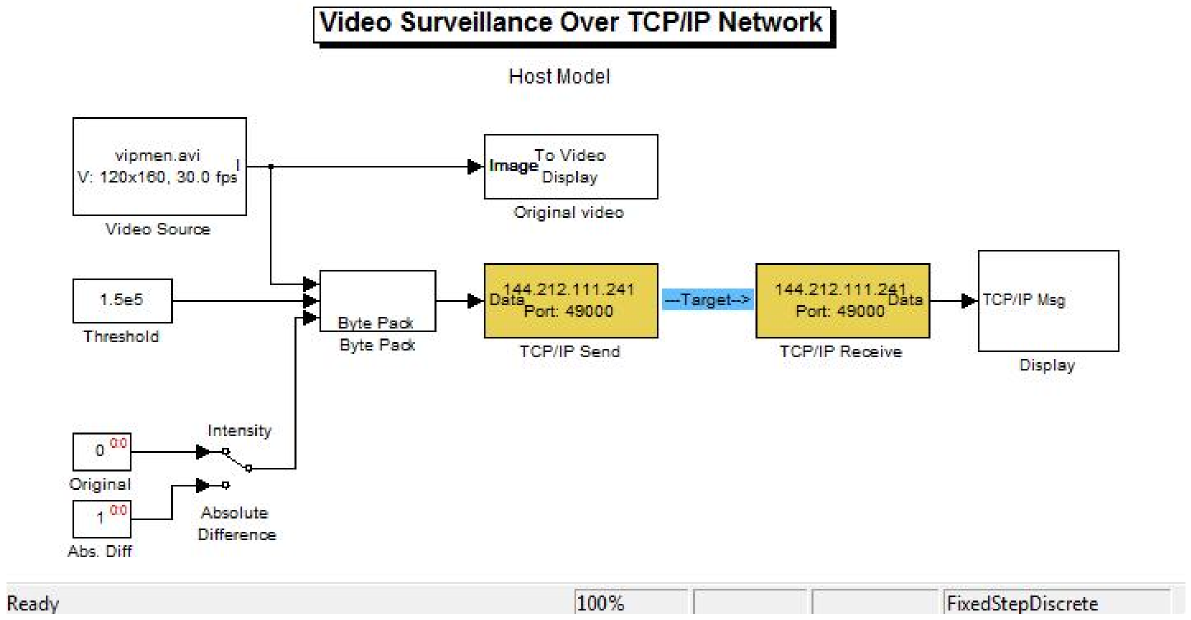

Figure 7: Camera Surveillance Using TCP/IP

The figure 7 above explains how camera surveillance are be done over a TCP/IP. The difference between this and the afore-mentioned is that this one requires a communication router to transmit video signals, while the later does not need a communication router. In other to minimize cost, the first approach is used to acquire live video feed.

The surveillance system was simulated using MATLAB/Simulink and deployed on the TI DSP platform. The host sends video frames via TCP/IP to the target. The target calculates SAD, compares it to the threshold, and returns the relevant frame if motion is detected.

Multi-camera input was synchronized using vertical sync pulses, and 3-D reconstruction was enabled through Bayer interpolation and storage in DDR memory.

Table 7: Comparative Results of Standalone SAD Surveillance and MASCOT-Integrated Design

Metric	**SAD Surveillance Alone**	**SAD + MASCOT Integration**
Modularity	Low (monolithic code)	High (well-defined modules)
Maintainability	Difficult due to tight coupling	Easy due to separation of concerns
Scalability	Limited to small setups	Easily expandable with new modules
Testing	Requires full system test	Enables unit and system-level testing

Performance	Real-time detection on DSP	Real-time detection with added system robustness
Synchronization	Manual implementation	Automated via dedicated sync module
Code Reuse	Low	High (modular components)

The results clearly showed that while the basic SAD system performs motion detection effectively, its real-world application is hindered by architectural rigidity. MASCOT integration significantly improves design discipline, traceability, and reusability. Using MASCOT enabled enhanced modularity for each system function, scalability to accommodate additional cameras or modules, simplified testing using independent module validation and improved maintainability and upgradeability

V. CONCLUSION

MASCOT is not a competing algorithm or technology for motion detection—it's a system design methodology. While your current SAD-based system focuses on real-time image processing and transmission, applying MASCOT would allow for better system modularity, scalability, and formalization, particularly beneficial if your system grows in complexity or needs to be deployed in mission-critical environments.
This work successfully demonstrates a scalable, modular, and intelligent surveillance system by combining a real-time 2-D SAD motion detection algorithm with MASCOT methodology. The proposed design improves system reliability and lays the groundwork for deploying intelligent surveillance in complex environments. Comparative analysis between SAD-based video surveillance motion detection system and the MASCOT methodology (Modular Approach to Software Construction Operation and Test), which is typically used in designing and documenting real-time, embedded, and distributed systems.

REFERENCES

E. J. Ong and S. Gong (2019), "Tracking Hybrid 2D-3D Human Models from Multiple Views", International Workshop on Modeling People at ICCV 2019, Corfu, Greece.

L. A. Hassnawi, R.B Ahmad, Abid Yahya, S. A. Aljunid, M. Elshaikh (2022), "Performance Analysis of Various Routing Protocols for Motorway Surveillance System Cameras' Network", IJCSI International Journal of Computer Science, Issues, Vol. 9, Issue 2, No 1, March 2022, ISSN (Online): 1694-0814, www.IJCSI.org

Maria Valera (2016), "An Approach for Real Time Distributed Intelligence Surveillance system", Amada Books, Ethiopia.

R. Rosales and S. Sclaroff (2020), "Inferring Body Pose Without Tracking Body Parts", Proceedings of Computer Vision and Pattern Recognition, South Carolina, pp. 721-727.