engineering proceedings

MDPI

Proceeding Paper

Real-Time Camera-Based Monitoring for Product Detection and Recognition †

Tsvetislava Lavchieva and Delyan Genkov *

Department of Computer Systems and Technologies, Technical University Gabrovo, 5300 Gabrovo, Bulgaria; tsvetislava.lavchieva@edu.mon.bg

* Correspondence: dgenkov@tugab.bg

† Presented at the International Conference on Electronics, Engineering Physics and Earth Science (EEPES2026), Bandirma, Turkey, 24–27 June 2026.

Abstract

This paper presents a prototype system for real-time camera-based monitoring of product presence and change events. The proposed approach combines object detection, class-based counting, and lightweight event logic to determine whether products appear, disappear, or change in quantity over time. The system is intended for live camera input, while the current implementation has also been tested in a controlled manner using prerecorded video sequences. A modular software structure is used to support future extensions, including experiments with additional detection models and product-specific training data. The current results show that the developed prototype can serve as a practical basis for intelligent visual monitoring in product observation scenarios.

Keywords: object detection; real-time monitoring; camera-based system; product presence detection; change event detection; computer vision

1. Introduction

Real-time visual monitoring has become increasingly important in applications where the observed scene changes continuously and requires timely interpretation. In many practical scenarios, it is not enough to detect objects in a single frame. It is also necessary to determine whether a specific product is present, whether it has disappeared, or whether its quantity has changed over time.

Camera-based monitoring provides a flexible and accessible way to address such tasks. Compared to manual observation, automated visual analysis can offer faster and more consistent information about the current state of the scene. This makes it relevant for applications related to product observation, inventory awareness, and dynamic scene analysis.

Recent advances in deep learning have significantly improved the performance of object detection methods in real-time environments. In particular, one-stage detectors have become widely used because they are capable of combining speed and detection accuracy in a way that is suitable for live visual processing. This has created opportunities for developing practical systems that can operate directly on camera streams and support higher-level monitoring functions [1].

At the same time, frame-level detection alone is often not sufficient in real monitoring tasks. A system may correctly recognize visible products, but this does not automatically provide information about changes in the scene. For many applications, the important question is not only what is currently visible, but also what has changed compared to the

Check for updates

Academic Editors: Teodor Iliev, Ivaylo Stoyanov, Grigor Mihaylov and Selahattin Kosunalp

Published: 8 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

previous state. This includes situations in which a product appears, disappears, or changes in quantity.

The present work addresses this problem through the development of a prototype system for real-time camera-based monitoring of product presence and change events. The proposed approach combines object detection, class-based counting, and state comparison in order to generate event-oriented information from visual observations. In this way, the monitoring process moves beyond isolated detections and toward a more meaningful interpretation of scene dynamics.

The current implementation is designed for live camera input, while controlled testing has also been carried out using prerecorded video sequences during development. The software structure is modular, which allows future extension of the system with additional detection models, custom training data, and broader experimental evaluation.

The main contribution of this paper is the design and implementation of a practical monitoring pipeline that integrates product detection with lightweight change-event logic in a real-time camera-based setting. The developed prototype is intended as a basis for further work on intelligent visual monitoring in product observation scenarios.

Recent developments in object detection show that real-time monitoring systems increasingly rely on efficient deep learning architectures capable of balancing detection accuracy and computational speed [1,2]. Modern YOLO-based detectors are widely used in practical applications because they provide a suitable compromise between performance and implementation complexity [3]. These developments support the motivation of the present prototype, which focuses on camera-based product monitoring and change-event detection in a controlled shelf-like environment.

2. Related Work

2.1. Real-Time Object Detection

Real-time object detection has developed rapidly in recent years due to the progress of deep learning methods. For practical monitoring systems, detection speed is as important as recognition accuracy, especially when the goal is to process a continuous visual stream from a camera [4,5]. For this reason, one-stage detectors have become a widely used choice in applications that require fast response and relatively lightweight processing. YOLO-based methods are particularly well known in this area because they are designed to balance speed and detection performance in real-time scenarios [1,6].

These developments have made object detection suitable not only for general visual recognition tasks, but also for applied monitoring systems. In such systems, the detector serves as the first stage of the processing pipeline by identifying the visible objects in each frame. However, in many real-world cases, detection alone is not enough. A monitoring system often has to interpret changes in the observed scene rather than simply list the objects that are currently visible [1].

Recent YOLO-related studies continue this direction by proposing architectural improvements intended to increase detection efficiency and reduce computational overhead [7,8].

2.2. Product Recognition and Shelf Monitoring

Product recognition in retail-like environments is more challenging than general object detection because many items have similar visual appearance, are partially occluded, or are observed under changing lighting and viewpoint conditions. Research on store-shelf analysis has shown that product-related recognition tasks often require handling a large number of visually similar classes, limited reference imagery, and frequent changes in packaging or product availability [2,3].

https://doi.org/10.3390/engproc2026154059

This makes product monitoring an especially relevant application domain for camera-based computer vision systems. Existing studies show that deep learning can support recognition and shelf analysis tasks in a way that is useful for automated store management, product observation, and inventory-related scenarios [4,9].

2.3. Change-Oriented Monitoring

In many monitoring applications, the most valuable information is not limited to what is present in a frame, but includes what has changed between consecutive observations. This idea is important in shelf monitoring and inventory-related settings, where the appearance of a new item, the disappearance of an existing one, or a change in visible quantity may be more useful than the raw detection output itself. Recent work on store monitoring pipelines reflects this shift toward change-aware scene interpretation and automated inventory observation [4,10].

The present work follows this direction, but focuses on a lightweight prototype that combines object detection with simple class-based counting and state comparison. Instead of aiming at a large-scale retail system, the goal is to build a practical camera-based monitoring pipeline that can identify product presence and generate basic change events in real time.

3. Materials and Methods

3.1. General System Concept

The proposed system is designed as a real-time monitoring pipeline based on camera input. Its main purpose is to observe a scene containing products and to determine changes in product presence over time. Instead of treating each frame as an isolated image, the system processes the video stream as a sequence of consecutive scene states and compares the current state with the previously stored one.

The overall workflow begins with frame acquisition from the camera input, followed by object detection and class-based counting. The detected products are summarized into a current scene state, which represents the number of objects from each recognized class at a given moment. This state is then compared with the previous scene state stored by the system. If a difference is identified, a change event is generated, such as the appearance, disappearance, or quantity change in a product.

After the comparison step, the current scene state becomes the previous state for the next processing cycle. In this way, the system forms a continuous monitoring loop rather than performing only independent object detection on separate frames. The output of the system includes both visualized detections and inventory-related information, allowing the user to observe the current product presence and the detected changes in the monitored scene.

The current concept is oriented toward live camera operation. At the present development stage, controlled tests with prerecorded video sequences have also been used to verify the logic of the pipeline and to observe the behavior of the system under repeatable conditions. This intermediate step supports the development process, while the intended final application remains real-time camera-based monitoring. Figure 1 illustrates the workflow of the proposed camera-based system for product monitoring and change detection.

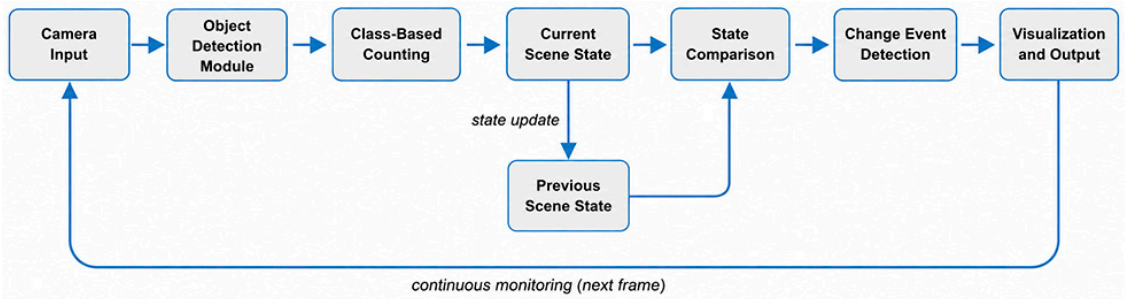


Figure 1. Workflow of the proposed camera-based system for product monitoring and change detection.

3.2. Software Environment and Implementation Structure

The prototype is implemented in Python v3.9 using a modular software structure [5]. The current version includes separate components responsible for configuration management, video stream handling, object detection, visualization, metrics support, recording of results, and inventory-related logic. This structure was selected in order to keep the system easier to organize, modify, and extend during further development.

A modular approach is especially useful in a prototype system that is expected to evolve. In the present case, it allows the monitoring logic to remain separate from the detection and visualization components. It also provides a practical basis for future experiments involving additional detection models, custom training data, and expanded evaluation procedures.

From an implementation perspective, the system is designed to support both controlled testing and direct transition to camera-based operation. This makes it possible to refine individual modules without changing the general architecture of the pipeline.

The object detection component used in the current prototype is based on the YOLOv8n model from the Ultralytics YOLOv8 family. This lightweight version was selected because it is suitable for real-time prototype experiments and camera-based testing under limited hardware resources. The model was initialized with pretrained yolov8n.pt weights and fine-tuned on a custom beverage-product dataset prepared for the purposes of the experimental scenario. The training configuration included 50 epochs, an input image size of 640 × 640 pixels, and a batch size of 16. The trained model is used in the prototype to localize visible products in each frame and to assign class labels that are later used for counting, state representation, and change-event detection.

3.3. Frame Processing and Detection

Each frame obtained from the selected video source is passed to the YOLOv8n-based detection stage of the system. The detection module identifies visible objects in the frame and associates them with class labels. These outputs form the basis for the current scene description. In the current prototype, the detection stage is implemented using the YOLOv8 model from the Ultralytics v8.4.142 framework [11].

The processed frame can also be visualized with annotations, which makes the system behavior easier to observe during execution. This annotated output is useful both for development and for practical monitoring, because it provides direct visual feedback about the detected products and their interpreted state.

At this stage, the detector is used as the primary mechanism for extracting object-level information from the scene. The following stages of the pipeline build on these outputs in order to derive more meaningful monitoring results. An example of the annotated visual output produced by the current prototype is shown in Figure 2.



Figure 2. Example of an annotated frame produced by the prototype system, showing detected beverage products and class-based inventory information.

3.4. Counting and State Representation

After the detection stage, the system counts the visible objects by class. This produces a compact representation of the current visual state of the observed scene. Instead of working only with raw detections, the pipeline transforms the frame-level output into structured count-based information.

Such a representation is useful because it simplifies the monitoring task. For example, if several products from the same category are visible, the system stores the number of detected instances for that class rather than treating each frame as an entirely new observation. In this way, the scene can be described through a set of current product counts.

This count-based representation also provides the necessary link between frame-level object recognition and change-aware monitoring. It allows the next stage of the system to compare consecutive states and determine whether a relevant event has occurred.

3.5. Change Event Logic

The main monitoring logic of the prototype is based on comparing the current scene state with the previously observed one. This makes it possible to generate event-oriented information rather than only a list of detected objects.

If a product class was not present in the previous state but is present in the current one, the system interprets this as an appearance event. If a previously present class is no longer detected, this is treated as a disappearance event. If the class remains visible but the number of detected instances changes, the system reports a quantity change event.

Although this logic is relatively simple, it is important because it shifts the monitoring process from static visual recognition toward basic scene interpretation. In practical terms, this means that the output becomes more informative and more suitable for observation tasks in which changes are more important than isolated detections. The basic interpretation logic used in the current prototype is summarized in Table 1.

Table 1. Basic event logic based on comparison between consecutive scene states.

Previous Count	Current Count	Interpreted Event
0	1	Appearance
1	0	Disappearance
2	1	Quantity decrease
1	2	Quantity increase
1	1	No change

As shown in Table 1, the event type depends on the difference between the previous and current count for each observed product class.

3.6. Current Test Scenario

The target application scenario of the developed system is real-time monitoring based on live camera input. In this setting, the camera continuously provides frames that are processed by the monitoring pipeline, allowing the system to observe changes in product presence as they occur.

At the current stage of development, controlled tests have also been performed with prerecorded video sequences. This approach is useful because it allows repeated observation of the same scene conditions and supports debugging of the counting and event-generation logic. It also makes it easier to verify whether the system responds consistently to known appearance and disappearance cases.

Even though prerecorded video has been used during development, the implemented architecture is intended for direct use in live camera-based monitoring. The present work therefore represents a prototype stage that bridges controlled validation and practical real-time operation.

4. Results

4.1. Current Prototype Capabilities

The current implementation shows that the proposed system is able to process video frames, detect visible products, count object instances by class, and generate event-related information based on state changes between observations. In addition to object-level recognition, the prototype provides a higher-level interpretation of the scene by identifying whether a product has appeared, disappeared, or changed in quantity.

The system also produces annotated visual output, which makes the monitoring process easier to follow during execution. This is useful both for development and for practical observation, since it allows the detected objects and the interpreted scene state to be inspected directly. In its current form, the prototype also supports the recording of output data, which can be used for later review and further analysis.

These results indicate that the developed solution is already functional as a monitoring prototype. Although the present stage does not yet represent a full experimental evaluation, it demonstrates that the core processing pipeline operates consistently and can serve as a basis for further refinement.

4.2. Example Monitoring and Change-Detection Cases

Several practical monitoring cases can be used to illustrate the current behavior of the prototype system. A basic case occurs when a product that was not present in the previous observation state becomes visible in the current frame sequence. In this situation, the system registers an appearance event. A second case occurs when a previously detected product is no longer visible, which leads to a disappearance event.

Another important case involves a change in the visible quantity of a given product class. For example, if two instances of a product are initially detected and only one remains visible in the following state, the system interprets this as a quantity decrease. In an analogous way, the addition of a newly visible instance is treated as a quantity increase.

These cases show that the proposed monitoring logic is capable of transforming frame-level detections into more meaningful scene-related outputs. Instead of providing only a static list of objects detected in a single frame, the prototype supports simple interpretation of changes over time. A representative example of a detected change event between two consecutive observation states is shown in Figure 3.

https://doi.org/10.3390/engproc2026154059



(**a**) (**b**)

Figure 3. Examples of change-event detection produced by the prototype system: (**a**) appearance of a Sprite product; (**b**) disappearance of a Sprite product.

4.3. Result Validation

The obtained results were validated through repeated practical tests with the developed prototype under controlled indoor conditions. The validation focused on three main aspects of the proposed monitoring pipeline: correct detection of the selected beverage product classes, correct formation of the class-based inventory representation, and correct identification of simple change events between consecutive observation states.

The test setup included a fixed camera position and a shelf-like scene containing beverage products from the trained classes. During the validation tests, products were added to and removed from the observed scene in order to verify whether the system correctly updated the inventory information and reported the corresponding appearance or disappearance event. The examples shown in Figure 3 illustrate representative successful cases in which the system detects the appearance and disappearance of a Sprite product and reflects the change in the inventory panel.

The validation was performed on a standard laptop (HP Inc., Palo Alto, CA, USA) configuration running Windows 11 Pro 64-bit, equipped with an Intel Core i3-1005G1 CPU (Inlel, Santa Clara, CA, USA) at 1.20 GHz, 8 GB of RAM (HP Inc., Palo Alto, CA, USA), and an NVIDIA GeForce MX330 graphics adapter with 2 GB of dedicated memory (Nvidia, Santa Clara, CA, USA). In the reported tests, the prototype was executed in CPU mode. During the tests, the system achieved a processing speed of approximately 4.8–6.3 FPS, as visible in the annotated output frames. Although this frame rate is not sufficient for high-speed industrial real-time applications, it was adequate for validating the proposed monitoring logic in a controlled shelf-like scenario, where the relevant product changes occur at relatively low speed.

In addition to the practical validation with camera-based test sequences, the trained detection model was evaluated on the validation subset of the prepared beverage-product dataset. The evaluation was performed after 50 training epochs using the standard object-detection metrics reported by the training framework. The obtained values are summarized in Table 2.

Table 2. Quantitative validation results of the trained YOLOv8 detection model.

Metric	Value
Precision	0.979
Recall	0.988
mAP@50	0.799
mAP@50–95	0.704
Validation box loss	0.767
Validation classification loss	0.932

The results show high precision and recall for the selected product classes, which indicates that the trained model is able to detect the target beverage products reliably under the controlled conditions used in the prototype tests. The mAP values provide an additional indication of the detection quality over the validation subset. These quantitative results support the qualitative observations from the live and prerecorded test sequences, although they should be interpreted within the limited scope of the current dataset and experimental setup.

The validation confirmed that the prototype is able to transform frame-level object detections into a compact inventory representation and to register basic changes in product presence over time. At the same time, the validation should be interpreted as prototype-level verification rather than as a complete large-scale benchmark evaluation. A broader quantitative evaluation with more test sequences, different lighting conditions, camera positions, and product arrangements is planned as part of the next development stage.

4.4. Practical Observations

The current implementation highlights several practical factors that influence system behavior. Detection quality depends on the visual conditions in the observed scene, including lighting, camera position, viewing angle, and partial occlusion of the products. In addition, visually similar packaging may increase the difficulty of correct class assignment.

Another important observation is that short-term instability in frame-level detections may lead to temporary fluctuations in the reported product count. For this reason, even a simple confirmation strategy across multiple frames can improve the stability of the generated monitoring events. This is particularly relevant in real-time operation, where transient detection errors may otherwise lead to incorrect appearance or disappearance reports.

These observations are important because they reflect realistic operating conditions rather than idealized examples. They also confirm that the development of such a monitoring system requires not only object detection, but also careful interpretation of detection results over time.

4.5. Limitations of the Current Stage

At the current stage, the developed system should be considered a functional prototype rather than a fully evaluated final solution. The present results are sufficient to demonstrate the viability of the proposed monitoring approach, but they do not yet constitute a broad comparative study. Additional work is still required in the areas of custom dataset creation, systematic live-camera testing, and controlled comparison between different detection models.

Another current limitation is that the experimental scope remains intentionally narrow. This is appropriate for the present development phase, since the main goal of this work is to establish the architecture and operating logic of the monitoring pipeline. Broader quantitative evaluation will be more meaningful after the next stage of implementation is completed. The current implementation status of the developed prototype is summarized in Table 3.

Table 3. Current functions of the developed prototype system.

Function	Current Status	Notes
Frame acquisition	Implemented	Supports video input and live camera transition
Object detection	Implemented	Based on the current detection module
Class-based counting	Implemented	Counts visible instances by class
Change event logic	Implemented	Supports appearance, disappearance, and quantity change
Live camera validation	In progress	Target application scenario
Multi-model comparison	Planned	Planned for future development

https://doi.org/10.3390/engproc2026154059

5. Discussion

The current prototype shows that combining object detection with simple state comparison can provide a practical basis for intelligent scene monitoring. This is important because, in many real-world applications, the useful information is not limited to the set of objects currently visible in a frame. In many cases, the more valuable information concerns the changes that occur over time, such as the appearance of a new product, the disappearance of an existing one, or a change in the visible quantity of a product class.

From this perspective, the proposed approach extends the role of object detection from frame-level recognition to basic scene interpretation. Even though the implemented event logic is relatively lightweight, it increases the practical usefulness of the system by transforming raw detections into monitoring-oriented outputs. This makes the prototype more suitable for observation tasks than a detector used in isolation.

Another important aspect of the current work is the modular structure of the system. The separation between frame acquisition, object detection, counting, state comparison, and visualization makes the prototype easier to expand in future development stages. This is particularly relevant for the planned integration of additional detection models, custom training data, and more structured evaluation under live camera conditions.

At the same time, the current prototype has limitations that should be acknowledged clearly. The system still depends on the quality and stability of frame-level detections, which means that lighting conditions, camera angle, partial occlusion, and visual similarity between products may affect the interpreted output. For this reason, a reliable monitoring system requires not only detection performance, but also stable temporal interpretation across consecutive observations.

The present stage should therefore be seen as a functional and promising basis rather than a final solution. Its main value lies in demonstrating that a relatively compact monitoring pipeline can move beyond isolated detections and support event-aware interpretation of product presence. This provides a suitable foundation for the next development stage, which will focus on live camera validation, product-specific training data, and broader experimental comparison.

6. Conclusions

This paper presented a prototype system for real-time camera-based monitoring of product presence and change events. The proposed approach combines object detection, class-based counting, and comparison of consecutive scene states in order to identify appearance, disappearance, and quantity changes.

The current implementation shows that such a pipeline can provide a practical basis for intelligent visual monitoring. Even at its present development stage, the system demonstrates the potential to move beyond frame-level object detection and toward more meaningful interpretation of scene dynamics.

At the same time, the work should be viewed as a prototype-stage contribution rather than a fully completed final solution. Its main value lies in establishing the architecture and operating logic of a camera-based monitoring system that can be extended in future development.

Future work will focus on live camera validation, creation of product-specific training data, integration of additional detection models, and broader experimental evaluation using suitable monitoring metrics.

Author Contributions: T.L. and D.G. were involved in the full process of producing this paper, including conceptualization, methodology, programming and preparing the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: The publication is supported by the Project № BG16RFPR002-1.014-0013-C01 "Digitalization of Economy in Big Data Environment—Second Stage" (DIGD2) financed by the "Research, Innovation and Digitalization for Smart Transformation" Program 2021–2027 and co-funded by the European Union.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors upon request.

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

References

1. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
2. Tonioni, A.; Serra, E.; Di Stefano, L. A Deep Learning Pipeline for Product Recognition on Store Shelves. In Proceedings of the 2018 IEEE International Conference on Image Processing, Applications and Systems (IPAS), Sophia Antipolis, France, 12–14 December 2018; pp. 25–31. [CrossRef]
3. Wei, Y.; Tran, S.N.; Xu, S.; Kang, B.H.; Springer, M. Deep Learning for Retail Product Recognition: Challenges and Techniques. *Comput. Intell. Neurosci.* **2020**, *2020*, 8875910. [CrossRef] [PubMed]
4. Pawar, S.; Jadhav, D.B.; Godse, D.; Jadhav, R.; Thakur, S. Vision-Based Empty Shelf Detection in Retail with Real-Time Telegram Notifications for Efficient Restocking. *SSRG Int. J. Electron. Commun. Eng.* **2024**, *11*, 180–187. [CrossRef]
5. OpenCV. Introduction—Open Source Computer Vision Library. Available online: https://docs.opencv.org/4.x/d1/dfb/intro.html (accessed on 10 April 2026).
6. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475. [CrossRef]
7. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv* **2024**, arXiv:2402.13616. [CrossRef]
8. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. *arXiv* **2024**, arXiv:2405.14458. [CrossRef]
9. Šikić, F.; Kalafatić, Z.; Subašić, M.; Lončarić, S. Enhanced Out-of-Stock Detection in Retail Shelf Images Based on Deep Learning. *Sensors* **2024**, *24*, 693. [CrossRef] [PubMed]
10. Rahi, B.; Sagmanli, D.; Oppong, F.; Pekaslan, D.; Triguero, I. Generalising Stock Detection in Retail Cabinets with Minimal Data Using a DenseNet and Vision Transformer Ensemble. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 66. [CrossRef]
11. Ultralytics. YOLOv8 Documentation. Available online: https://docs.ultralytics.com/models/yolov8/ (accessed on 10 April 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.