OBJECT MOTION DETECTION

Submitted To -	**Prof. Balaji Vijaykumar**
Submitted By-	**Deepankar Gupta (4NI23CI033)**

Git Repository of report →

deepankargupta856/Object-motion-detection

DATE: 16th April 2025

AIM: Detect whether an object in frame is moving or not.

ALGORITHM:

1.Detect the object in frame **(OBJECT DETECTION)**

2.Decide whether in motion or not. (**MOTION DETECTION)**

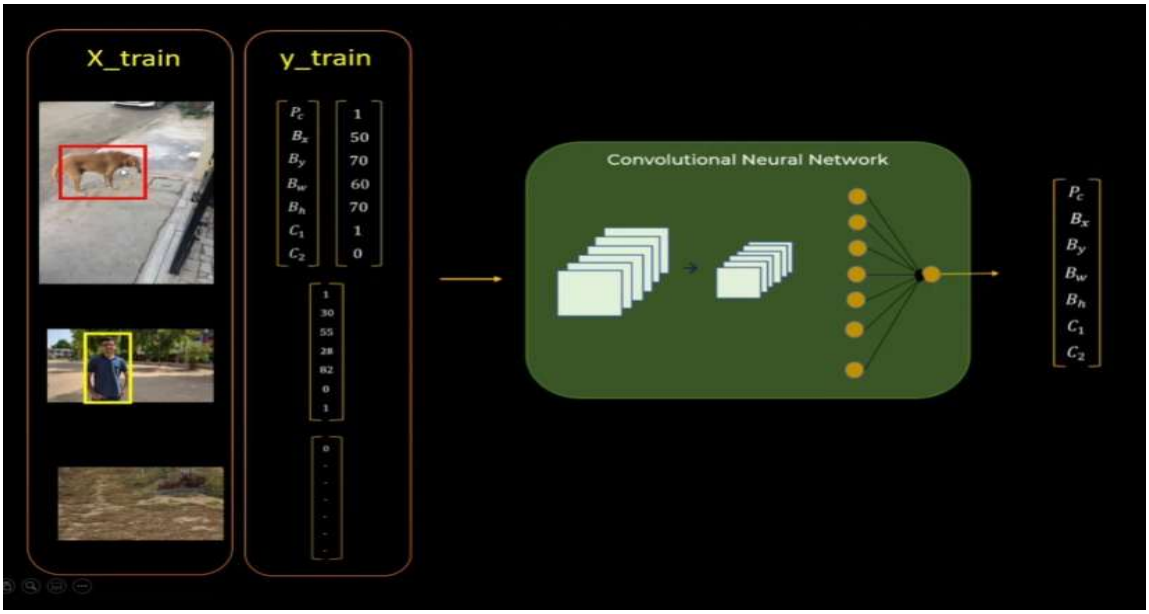
OBJECT DETECTION –

It's a computer vision technique that identifies and locates objects in frames.

It answers three key questions:

1. What is in the image? -> Class label *(e.g., cat, car)* detection and classification.
2. Where is it? -> Bounding box around *the object* ,localizing.
3. How sure are we? -> Confidence score *(e.g., 95% sure it's a dog)*

object detection = **classification + localization**.



In the above vector's :

Pc -> detects whether object is present (0 for absence 1 for absence)

Bx, By, Bw, Bh -> bounding box coordinates

C1, C2, … , Cn-> different class representing different entities in frame

Traditional Methods Pre-Deep Learning era –

1. **Haar Cascades –** A machine learning based approach for object detection, introduced in seminal paper "Robust Real-Time Face Detection".
   About paper – the research paper proposed a framework to detect faces in images with high speed and accuracy.

   The key concepts introduced were, speeding up feature evaluation, using simple but informative features, intelligently selecting the best features and building a strong classifier, and efficiently focusing computation on relevant image regions to achieve robust real-time face detection.
2. **HOG (Histogram of Oriented Gradients)** + **SVM –** HOG was used to extract features from image and SVM for classification.

These methods required **manual tuning**, and couldn't handle multiple or overlapping objects well.

Deep Learning Era: Two-Stage Detectors – revolutionized object detection by using Convolutional Neural Networks to automatically learn features from data.

1. R-CNN (Region-based CNN, 2014) – use selective search to find possible object regions (region proposals) then run a CNN on each region, using SVM for classification. Accurate, but very slow as each region is processed independently.
2. Fast R-CNN (2015) – Improves R-CNN by processing the entire image once with a CNN, then extracts features from region proposals using RoI Pooling. Much faster than R-CNN.

3. Faster R-CNN (2015) – Introduced the Region Proposal Network (RPN), that merges region proposal generation and classification into a single unified network. Much faster, still very accurate, widely used in medical imaging, autonomous vehicles, and surveillance.

Two-stage detectors prioritize accuracy, even if it costs some speed.

Real-Time Object Detection – One-Stage Detectors skip the region proposal step, treating object detection as a single regression problem.

1. YOLO (You Only Look Once) – YOLO divides the input image into an **S×S grid**, where each cell predicts **bounding boxes**, their **confidence scores**, and **class probabilities** if an object's centre lies in that cell. This turns detection into a **single regression problem**, allowing the model to detect multiple objects in one pass. Its speed and efficiency make YOLO ideal for **real-time applications** like self-driving cars and surveillance.
   Evolved significantly:
   YOLOv1 – Basic version.
   YOLOv3/YOLOv4 – Improved accuracy.
   YOLOv5/YOLOv8 – Modular, production-ready, PyTorch-based with easy APIs.
2. SSD (Single Shot Multi-box Detector) – Detects objects at **multiple scales** using feature maps from different CNN layers, balances **speed and accuracy**. Better at detecting **small objects** than early YOLO versions.
3. Retina-Net – Introduced Focal Loss to solve class imbalance (common objects dominate rare ones). High accuracy, particularly for **dense scenes** (e.g., crowd detection).

One-stage detectors are used in **mobile apps, AR, drones, and robotics** because of their speed.

Transformer-Based Models (2020–Present) – originally designed for NLP, were later adapted for vision tasks especially object detection.

1. **DETR (Detection Transformer, 2020)**
    - Developed by Facebook AI.
    - Uses transformer encoders and decoders to detect objects without region proposals or anchor boxes.
    - **Drawbacks**: Slower to train, struggles with small objects.
2. **DINO, ViTDet, Sparse R-CNN**
    - Build upon DETR and vision transformers.
    - More sample-efficient, scalable, and better at handling complex scenes.
    - Often used in research and high-end applications.

Transformers allow long-range dependency modelling, enabling the network to understand context and relationships between objects.

There is an issue of incorrect object predictions in YOLO, such as misclassifying a car as a suitcase or cell phone, which is mainly due to low-resolution inputs and visual similarities between classes. The pre-trained model used is limited to 80 COCO classes, which may not cover all real-world scenarios accurately. A low confidence threshold and default anchor boxes not suited to the object's size or shape can also lead to inaccurate results.

To improve accuracy, the confidence threshold can be raised to filter out weak predictions, and higher-quality video input should be used. Switching to a larger model like YOLOv5m, fine-tuning on a custom dataset, applying post-processing filters, and adjusting anchor boxes can significantly enhance detection reliability and reduce misclassifications.

MOTION DETECTION –

It refers to the process of detecting changes in an environment, typically in video frames over time, to identify areas where an object has moved or is moving, which can be crucial in applications like:

- Security: Surveillance systems detecting intruders.
- Autonomous Vehicles: Detecting pedestrians or other vehicles in motion.
- Healthcare: Monitoring patients or elderly individuals for irregular movements.
- Robotics: Enabling robots to track and interact with moving objects.

Evolution of Motion Detection Technology

Early phase – They depended on observing changes using various physical metrics.

1. Image pixel-based detection → based on analysing the change in pixels in sequential frames.
2. Radar based motion sensors → used electromagnetic waves to capture the any objects interfering with the wave path.

Current Landscape – with increase in computation power and smaller chips we have made significant improvements in motion detection technology.

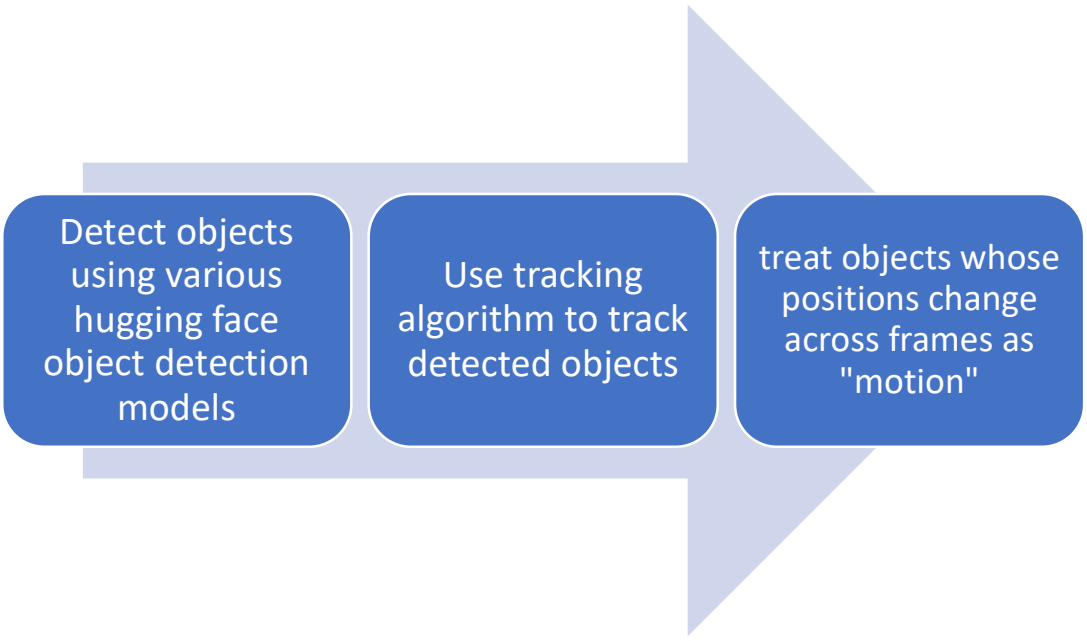
1. Computer vision and AI based motion detection → uses Deep learning.
2. LiDAR and 3d motion tracking → Light Detection and Ranging is a remote sensing technology that uses laser pulses to measure distances and create detailed 3D maps.
3. Smart IoT integration → development advanced motion sensor chips due to miniaturization and manufacturing improvements.

To achieve true motion detection, you'll typically need to:

- Process a sequence of frames from a video.
- Use an object detection model to identify objects in each frame.
- Track the position of objects across frames to determine their movement.
- Libraries such as OpenCV are very useful for the tracking portion of this task.

In my search I found there aren't models specifically for motion detection, we can use an object detection model to firstly identify object(s) then we can pair with motion tracking algorithms like ex. DeepSORT, ByteTrack etc.

A flow for motion detection →



We can also use OpenCV for classic motion detection:

- cv2.absdiff() → uses frame differencing method (Frame differencing works by comparing two consecutive frames of a video. If there's a significant change in pixel values between the two, it's considered as motion, its con is that it fails with slow-moving objects or variable lighting.)
- cv2.createBackgroundSubtractorMOG2() → uses background modelling (Background subtraction helps the computer learn what the usual scene looks like using statistical methods [Gaussian mixture model, KNN], and then spot anything new or different, it struggles with moving backgrounds but has higher accuracy than frame differencing)

These are lightweight**,** non-deep-learning methods which are effective for CCTV, and other simple applications.

Frame differencing and background subtraction algorithm →

implementation code

1. Import necessary libraries
2. Load the video
3. Initialize background subtractor (MOG2)
4. Read initial video frames
5. Loop over frames
6. Apply frame differencing
7. Find and analyse contours
8. Apply background subtraction (MOG2)
9. Display and print motion status, prepare for the next frame, repeat until all frames processed or limit reached

Motion Detection and Tracking using Deep Learning Models and Tracking Algorithms

1. Object Detection using Hugging Face Models

In this step, we use pre-trained object detection models from Hugging Face, such as facebook/detr-resnet-50, hustvl/yolos-small, or IDEA-Research/dino-vitb16, to identify and localize objects within video frames. These models work by processing the frame and extract spatial features which are then passed through a detection head that predicts:

- Bounding boxes (coordinates defining the object's location)
- Class labels (e.g., car, person, dog)
- Confidence scores (probability of correct classification)

The result is the detection of multiple objects with their exact positions and class types in each frame (implementation in object detection section)

2. Object Tracking Across Frames

After detecting objects, the next challenge is tracking these objects across multiple frames to establish continuous object identities, which can be accomplished by using tracking algorithms, including:

- **SORT** (Simple Online and Realtime Tracking): A fast, Kalman filter-based algorithm that predicts object movement and links detections between frames. It relies on spatial location but doesn't use object appearance.
- **Deep SORT**: An enhancement over SORT that integrates deep learning-based appearance features, enabling better identity persistence and handling occlusions more effectively.

These trackers assign unique IDs to each object and maintain consistent tracking over time.

3. Motion Estimation via Positional Change

By combining detection and tracking, the movement of objects is determined by analysing changes in their positions over time. When the bounding box of a tracked object shifts between frames, it indicates motion. This method provides a more robust form of motion detection compared to traditional techniques like frame differencing or background subtraction, as it focuses on the movement of specific, identified objects rather than background noise. The integration of object classification and motion tracking ensures the system can identify and track motion across different objects with high accuracy.

Tracking Algorithms →

SORT (Simple Online and Realtime Tracking)

SORT is a real-time object tracking algorithm that links detected objects across video frames using spatial information. It operates using:

1. **Kalman Filter** – A mathematical estimator that predicts an object's next position using its previous trajectory, accounting for noise and uncertainty. It has a two-step process: prediction (estimating the next location) and update (adjusting based on the actual detection).
2. **Hungarian Algorithm** – An optimization method that assigns detections to existing tracked objects by minimizing a cost function, typically using IoU (Intersection over Union). IoU measures the overlap between predicted and detected bounding boxes, guiding the best possible assignment.

While SORT is fast and efficient, it only considers positional data and lacks performance in complex scenarios with occlusions or similar-looking objects.

Deep SORT

Deep SORT builds upon SORT by incorporating appearance-based information for better identity tracking. It includes:

1. **Feature Extraction Network** – A pre-trained Convolutional Neural Network (CNN) that extracts feature vectors from detected object images. These vectors represent visual characteristics like colour and shape, helping to distinguish objects.
2. **Cosine Similarity + IoU(**intersection over union) – Deep SORT uses both cosine similarity (to compare visual appearance via feature vectors) and IoU (to assess spatial overlap) for assigning detections to tracks. Cosine similarity calculates how similar two appearance vectors are, enabling the tracker to handle occlusions and re-identify objects more accurately.

Deep SORT is more robust in crowded scenes and offers improved performance by using both motion and appearance cues.

Designing a flow for the motion detection algorithm →

We need to create a pipeline which is designed to detect and track multiple objects across video frames in real-time by combining YOLOv5 for fast and accurate object detection with DeepSORT to track the identities of detected objects across video frames. It enables not just identifying what is present in a scene, but also tracking their movement and continuity, which is essential for applications like surveillance, traffic monitoring, and behaviour analysis.

Algorithm for processing :

1. **Upload and load the video**
2. **Load the YOLOv5 model**
3. **Initialize the Deep SORT tracker**
4. **Loop over video frames:**
    - Capture the current frame
    - Detect objects using YOLOv5
    - Extract bounding boxes, class IDs, and confidence scores
    - Format detections for Deep SORT
    - Update tracker with current detections
    - Retrieve and assign unique track IDs
    - Annotate the frame with bounding boxes and IDs
    - Save the annotated frame to the output video
5. **Finalize and save the output video**
6. **Display or download the result**

Reference for implementation –

Computer-Vision-Projects/Yolo_V11_object_detection/yolov11.ipynb at main · Gagan824/Computer-Vision-Projects