Abstract

In recent years, fire accidents have posed serious threats to both human life and property. Traditional fire detection systems, such as smoke detectors or thermal sensors, often suffer from delayed response times and high installation costs. This project proposes a real-time fire and smoke detection system using computer vision techniques implemented with OpenCV. The system utilizes video feed input from a camera and applies image processing methods to detect fire and smoke based on color, motion, and texture analysis.

Communication barriers faced by the deaf and hard-of-hearing community can be significantly reduced through real-time sign language translation systems. The system captures hand movements through a webcam and processes the input using image processing techniques such as background subtraction, skin color segmentation, and contour detection. Key features like finger positions, hand orientation, and gesture dynamics are extracted and classified using machine learning models integrated with OpenCV.

Ensuring food quality is a critical aspect of agricultural production, directly impacting consumer health, market value, and supply chain efficiency. Traditional food quality inspection methods are often labor-intensive, subjective, and time-consuming. With advancements in computer vision and artificial intelligence, automated inspection systems offer a promising alternative.

Introduction

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. Initially developed by Intel, OpenCV provides a rich set of tools to handle a wide range of tasks related to computer vision, image processing, machine learning, and artificial intelligence. Its primary goal is to make real-time computer vision and image analysis more accessible, particularly for developers and researchers.

1.Real-Time Fire and Smoke Detection -Real-time fire and smoke detection systems are critical technologies designed to identify fire-related hazards as early as possible to minimize damage and ensure safety.

Sign Language Translation Tool -This tool uses computer vision, machine learning, and natural language processing to detect and interpret sign language gestures in real-time, translating them into spoken or written language—and vice versa.

Food Quality Inspection in Agriculture - Food Quality Inspection in Agriculture involves evaluating various attributes such as appearance, size, color, texture, ripeness, and the presence of contaminants or defects.

OpenCV

Aim:

The aim of this project is to design and implement three computer vision-based systems using OpenCV: Real-Time Fire and Smoke Detection, a Sign Language Translation Tool, and a Food Quality Inspection system for agriculture. These applications utilize image processing and real-time video analysis to enhance safety, communication, and agricultural quality control. By leveraging the capabilities of OpenCV, the goal is to develop efficient, cost-effective, and accurate solutions that address real-world problems in diverse domains

1. **Real-Time Fire and Smoke Detection**

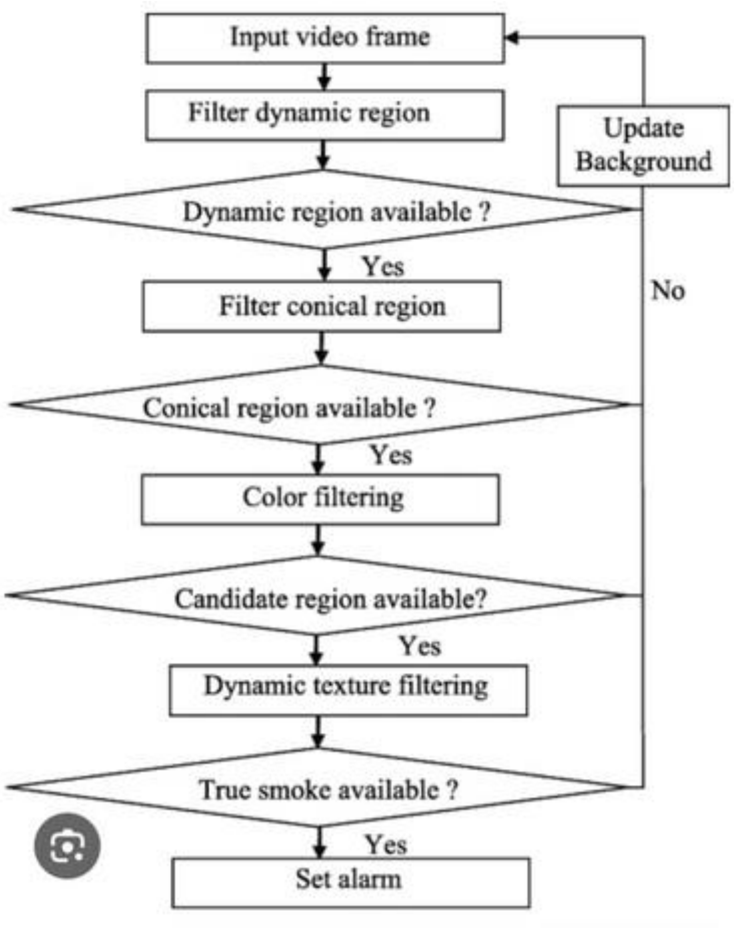
   Description:

   Real-time fire and smoke detection is a computer vision-based technique that continuously monitors video streams to identify fire or smoke as early warning signs of fire hazards. The system analyzes each video frame as it is captured, allowing quick detection and response.

   Design Strategy:

   - Capture video from cameras or other sources.
   - Use OpenCV's image processing algorithms to detect fire and smoke (e.g., color-based detection, motion analysis).
   - Train a machine learning model (e.g., SVM, deep learning) to classify the detected regions as fire or smoke.
   - Send alerts and notifications to authorities or emergency services when fire or smoke is detected.

Block diagram:



Program:

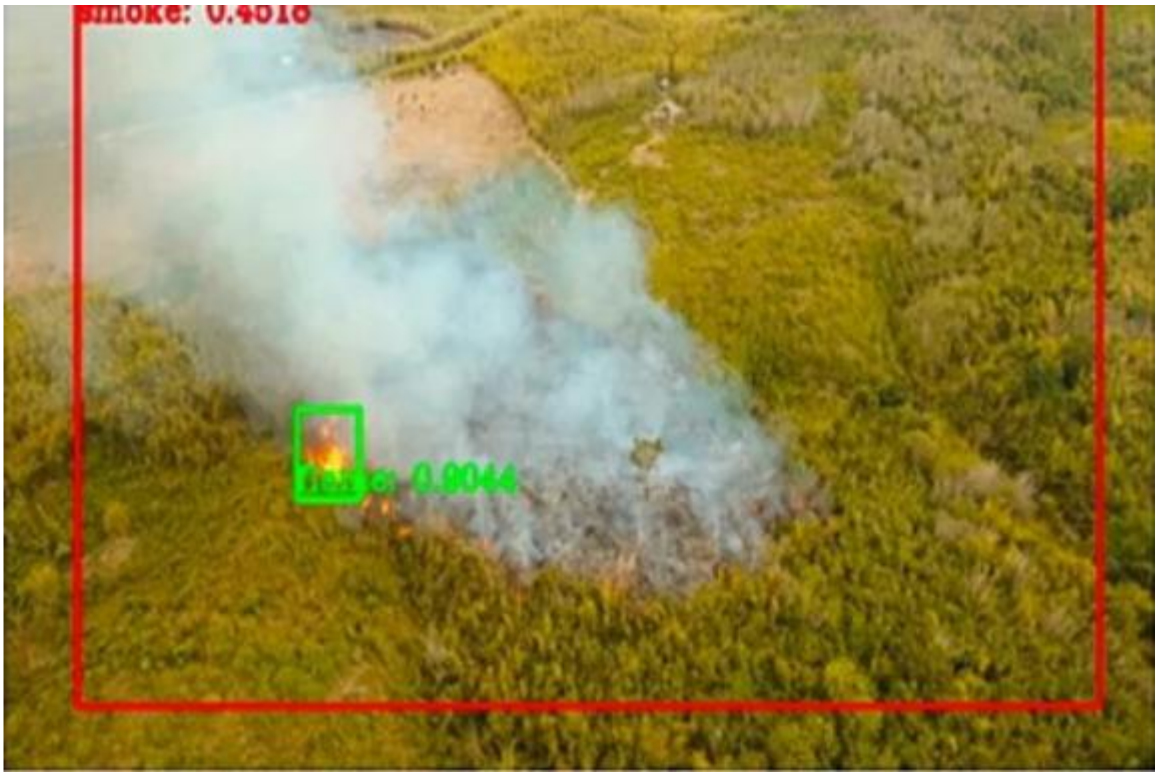
```
import cv2

import numpy as np


cap = cv2.VideoCapture(0)


ret, prev_frame = cap.read()

prev_frame = cv2.GaussianBlur(prev_frame, (21, 21), 0)

prev_gray = cv2.cvtColor(prev_frame, cv2.COLOR_BGR2GRAY)
```

```
while True:
    ret, frame = cap.read()
    if not ret:
        break

    blurred = cv2.GaussianBlur(frame, (21, 21), 0)
    gray = cv2.cvtColor(blurred, cv2.COLOR_BGR2GRAY)

    diff = cv2.absdiff(prev_gray, gray)
    _, thresh = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)

    thresh = cv2.dilate(thresh, None, iterations=2)

    contours, _ = cv2.findContours(thresh.copy(),
cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        area = cv2.contourArea(contour)
        if area > 5000:  # Fire tends to have chaotic large-area movement
            (x, y, w, h) = cv2.boundingRect(contour)
            roi = frame[y:y+h, x:x+w]
            roi_hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
            mask = cv2.inRange(roi_hsv, (0, 50, 50), (35, 255, 255))  # basic fire color mask

            fire_pixels = cv2.countNonZero(mask)
            if fire_pixels / (w * h) > 0.4:  # enough fire-colored pixels in motion area
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
                cv2.putText(frame, "FIRE!", (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0, 255), 2)

    cv2.imshow("Fire Detection", frame)
    prev_gray = gray.copy()

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()

import cv2
import numpy as np

def detect_fire_and_smoke(image_path):
    image = cv2.imread(image_path)
    if image is None:
        print("Image not found or path is incorrect.")
        return

    output = image.copy()
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

    fire_lower = np.array([0, 50, 200])
    fire_upper = np.array([35, 255, 255])
    fire_mask = cv2.inRange(hsv, fire_lower, fire_upper)

    fire_mask = cv2.morphologyEx(fire_mask, cv2.MORPH_OPEN,
np.ones((5,5), np.uint8))

    contours, _ = cv2.findContours(fire_mask, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if area > 1000:
            x, y, w, h = cv2.boundingRect(cnt)
            cv2.rectangle(output, (x, y), (x+w, y+h), (0, 0, 255), 2)
            cv2.putText(output, "FIRE", (x, y-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0,0,255), 2)
```

```
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (21, 21), 0)

    laplacian = cv2.Laplacian(blurred, cv2.CV_64F)
    contrast = np.var(laplacian)

    if contrast < 50:  # Empirical threshold
        cv2.putText(output, "SMOKE LIKELY DETECTED", (10, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 255, 0), 2)

    cv2.imshow("Detection", output)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

detect_fire_and_smoke("fire.png")
```

Input

Output:



Application:

1. Industrial settings: Factories, warehouses, and manufacturing facilities.

2. Public spaces: Malls, airports, and train stations.

3. Residential areas: Homes and apartments.

2.Sign language transition tool

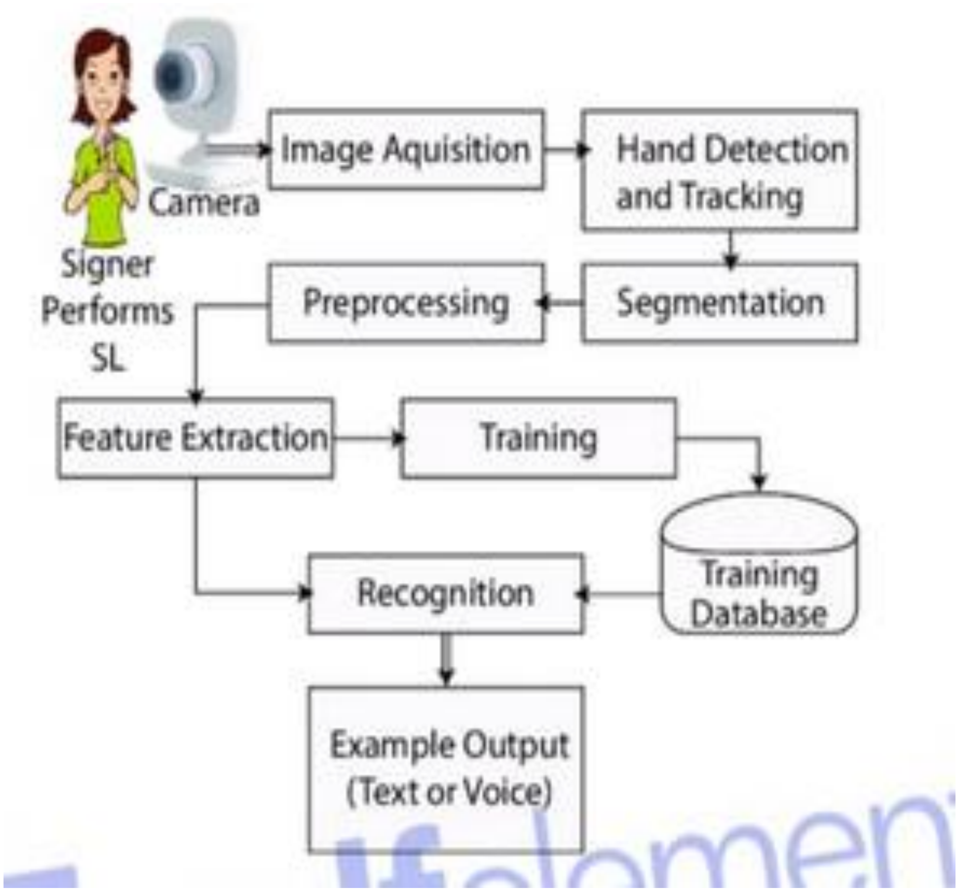
Description:

The Sign Language Translation Tool is a computer vision-based application that leverages OpenCV and machine learning to recognize and translate hand gestures from sign language into readable or spoken text. This tool aims to bridge communication gaps between the hearing-impaired and those unfamiliar with sign language.

Design strategy:

- Collect dataset of sign language gestures.
- Train machine learning model using collected dataset.
- Use OpenCV for real-time hand detection and gesture recognition.
- Integrate with text-to-speech system or display.

Block diagram:



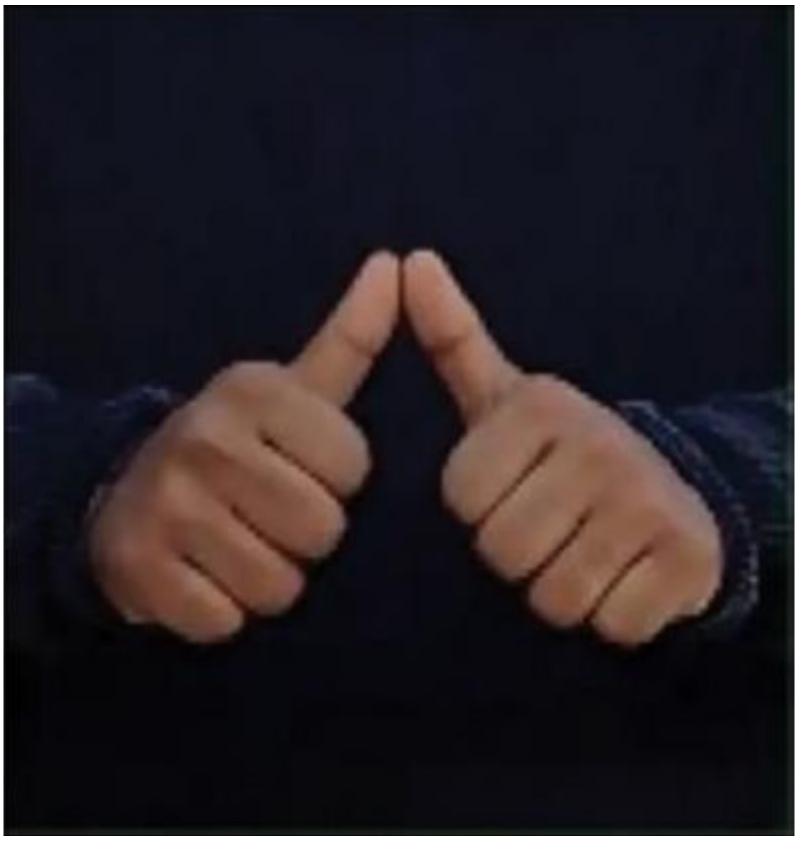
Program:

```
import cv2

import numpy as np


def get_hand_gesture(contour, drawing):

    hull = cv2.convexHull(contour, returnPoints=False)

    if len(hull) > 3:

        defects = cv2.convexityDefects(contour, hull)
    if defects is not None:
        count_defects = 0
        for i in range(defects.shape[0]):
            s, e, f, d = defects[i, 0]
            start = tuple(contour[s][0])
            end = tuple(contour[e][0])
            far = tuple(contour[f][0])

            a = np.linalg.norm(np.array(start) - np.array(end))
            b = np.linalg.norm(np.array(start) - np.array(far))
            c = np.linalg.norm(np.array(end) - np.array(far))
            angle = np.arccos((b*2 + c2 - a*2) / (2 * b * c + 1e-5))

            if angle <= np.pi / 2:  # 90 degree
                count_defects += 1
                cv2.circle(drawing, far, 5, [0, 0, 255], -1)
        if count_defects >= 4:
            return "Open Hand"
        elif count_defects == 0:
            return "Fist"
        else:
            return "Gesture"
return "Unknown"
```

```
cap = cv2.VideoCapture(0)


while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv2.flip(frame, 1)


    roi = frame[100:400, 100:400]
    cv2.rectangle(frame, (100, 100), (400, 400), (0, 255, 0), 2)


    hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)


        lower_skin = np.array([0, 20, 70], dtype=np.uint8)
    upper_skin = np.array([20, 255, 255], dtype=np.uint8)


    mask = cv2.inRange(hsv, lower_skin, upper_skin)
    mask = cv2.GaussianBlur(mask, (5, 5), 100)


    contours, _ = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
```

```
    if contours and len(contours) > 0:
        contour = max(contours, key=cv2.contourArea)
        drawing = np.zeros(roi.shape, np.uint8)
        cv2.drawContours(drawing, [contour], 0, (0, 255, 0), 2)

        gesture = get_hand_gesture(contour, drawing)
        cv2.putText(frame, f"Gesture: {gesture}", (50, 50),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 2)

        if gesture == "Open Hand":
            cv2.putText(frame, "Hi", (50, 100),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255), 3)

        drawing_resized = cv2.resize(drawing, (300, 300))
        frame_height, frame_width = frame.shape[:2]
        if frame_width >= 750:
            frame[100:400, 450:750] = drawing_resized
        else:
            frame[100:400, frame_width-300:frame_width] =
drawing_resized

    cv2.imshow("Sign Language Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()

cv2.destroyAllWindows()
```

Input:



Output:

Application:

1.Real-Time Translation: Translate sign language into spoken language in real-time.

2. **Hand Detection and Tracking:** Use OpenCV to detect and track hand movements.

3. **Gesture Recognition:** Recognize sign language gestures using machine learning models.

4. **Text-to-Speech:** Convert translated text into speech.

5. **User Interface**: Provide a user-friendly interface to interact with the application.

3.Food quality inspection in Agriculture

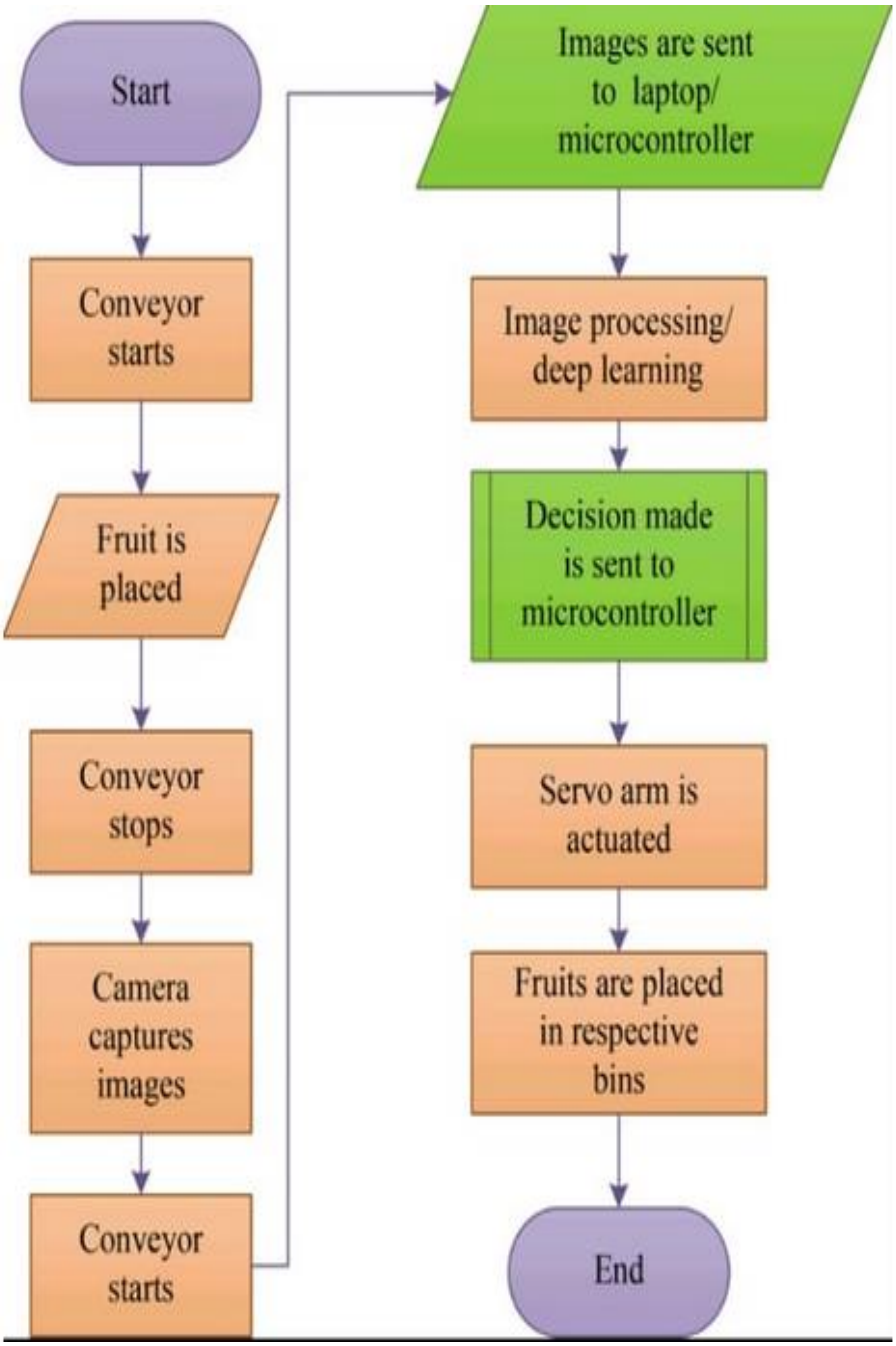
Description:

Food Quality Inspection in Agriculture using OpenCV involves the use of computer vision techniques to automatically assess the appearance and condition of agricultural produce. The system uses cameras to capture images of the produce, and OpenCV processes these images to identify defects, classify products, and ensure compliance with quality standards.

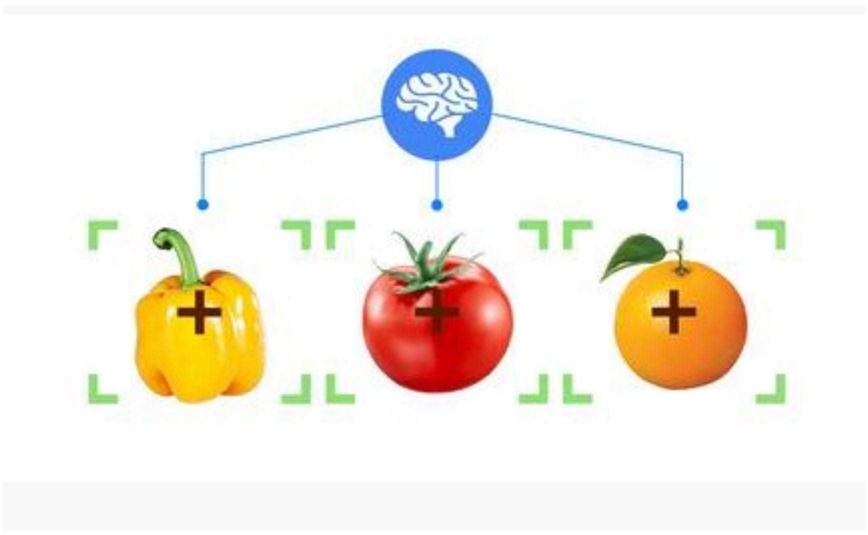
Design strategy:

- Take photos of food products.
- Use computer vision to detect defects.
- Use machine learning to classify products as good or bad.
- Integrate image analysis and machine learning for automated inspection.

Block diagram:


Start
Conveyor starts
Fruit is placed
Conveyor stops
Camera captures images
Conveyor starts
Images are sent to laptop/ microcontroller
Image processing/ deep learning
Decision made is sent to microcontroller
Servo arm is actuated
Fruits are placed in respective bins
End

Output:



Application:

1. Fresh Produce Inspection: Inspecting fruits and vegetables for quality, safety, and freshness.

2. Grain Quality Assessment: Evaluating grains for quality, moisture content, and contamination.

3. Dairy and Meat Inspection: Inspecting dairy products and meat for quality, safety, and authenticity.

4. Packaged Food Inspection: Verifying packaged food products for quality, labeling, and compliance.

5. Export and Import Inspection: Ensuring food products meet international quality **and** safety standards.

Result:

A system that detects fires and smoke in real-time, enabling prompt action to prevent damage. A tool that translates sign language into text or speech, facilitating communication between sign language users and others. A system that inspects food products for quality and defects, ensuring safe and healthy food supply.