**Technical Report: Plastic Sorting Using Object Detection and Image Processing Techniques**

**Authors:** Sarath Chandran, M Devanarayanan Namboothiri

**Abstract**

This report presents a comprehensive methodology for sorting plastic materials using image processing and object detection techniques implemented via OpenCV. The objective is to automate the plastic sorting process by detecting and classifying plastic objects on a moving conveyor belt based on size and color characteristics. We discuss preprocessing methods, background subtraction, contour analysis, and propose potential AI/ML-based improvements for real-world applicability.

**1. Introduction**

Plastic waste management is a growing concern requiring efficient sorting systems. Manual sorting is labor-intensive and inconsistent. Automating the sorting process using computer vision offers a promising solution. This project explores techniques in image acquisition, processing, and object analysis using OpenCV for real-time plastic sorting.

**2. Frames: Video Representation in OpenCV**

OpenCV handles video as a sequence of frames, with each frame represented as a 2D NumPy array. For a 640x480 RGB frame, the array shape is (480, 640, 3), representing height, width, and color channels (Red, Green, Blue).

Grayscale frames reduce the shape to (480, 640) by converting RGB to a single intensity value using:

Y = 0.299*R + 0.587*G + 0.114\*B

This transformation simplifies processing and reduces resource consumption.

**3. Image Enhancement Techniques**

**3.1 Gaussian Blur**

A smoothing technique to reduce image noise. Implemented via:

cv2.GaussianBlur(image, (5,5), 0)

A Gaussian kernel is convolved over the image to smoothen pixel intensity variations.

**3.2 Morphological Opening**

A sequence of erosion followed by dilation:

Opening(A) = Dilation(Erosion(A)) = A⊖B ⊕ B

* Erosion removes small noise and thins objects.
* Dilation restores the object size while excluding noise.

**4. Thresholding and Binarization**

Used to convert grayscale images to binary using a threshold value:

\_, binary\_image = cv2.threshold(gray\_image, 75, 255, cv2.THRESH\_BINARY)

Pixels above 75 become white (255), others become black (0), simplifying object detection.

**5. Contour Detection and Analysis**

Contours represent continuous boundaries of white regions in a binary image. OpenCV uses the Suzuki-Abe algorithm for contour detection:

contours, \_ = cv2.findContours(binary\_image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

Contours are analyzed for:

* **Color**: Using masks drawn over contours to extract mean RGB/HSV values.
* **Size**: Using cv2.contourArea(contour) to calculate area in pixels, which is calibrated to real-world units.

**6. Plastic Sorting Workflow**

1. Grayscale conversion of the frame.
2. Apply threshold to create binary image.
3. Detect contours.
4. Analyze color and size.
5. Classify plastic based on predefined rules.

**7. Background Subtraction Techniques**

**7.1 Static Background Mean Method**

1. Capture 60-120 empty conveyor frames.
2. Calculate mean background:

mean\_bg = numpy.mean(stored\_frames, axis=0)

1. Compare current frame with mean background to isolate moving objects.

**7.2 Video Subtraction**

Use pre-recorded video of empty conveyor synchronized with real-time feed for background subtraction. This addresses stitching and alignment issues.

**8. Object Validation Criteria**

**Stability Threshold**: A contour must persist across multiple frames to be considered valid.

**Area Threshold**: Only contours exceeding a minimum area are analyzed.

**9. Evaluation of Methods**

**Strengths:**

* Accurate detection under controlled lighting.
* Real-time processing supported.
* Adaptable to various conveyor speeds.

**Limitations:**

* Sensitive to lighting changes.
* Synchronization issues in video subtraction.
* Processing limitations at high speeds.

**10. Future Work**

**AI/ML Integration**:

* Employ deep learning models (e.g., YOLO, Ultralytics) for object detection.

**Hardware Optimization**:

* Integrate with NIR sensors and mechanical actuators.
* Improve conveyor calibration to mitigate stitching artifacts.

**Software Enhancements**:

* Explore dynamic background subtraction techniques from OpenCV.
* Optimize synchronization algorithms.

**References**

1. <https://pysource.com/2020/05/19/identify-objects-moving-on-a-conveyor-belt-using-opencv-with-python/>
2. <https://docs.opencv.org>
3. Bergstrom, A. C., et al., Gaussian Blur and Edge Response, 2023
4. Maragos, P., & Pessia, L., Morphological Filtering for Image Enhancement
5. Suzuki, S., & Abe, K., Border Following in Binary Images, 1985
6. <https://docs.ultralytics.com>

**Appendix: Sample Code Snippets**

# Gaussian Blur  
blurred = cv2.GaussianBlur(gray, (5,5), 0)  
  
# Thresholding  
\_, binary = cv2.threshold(blurred, 75, 255, cv2.THRESH\_BINARY)  
  
# Find Contours  
contours, \_ = cv2.findContours(binary, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
  
# Contour Area and Filtering  
valid = [cnt for cnt in contours if cv2.contourArea(cnt) >= 500]

**End of Report**