Car Component Dataset Creation

Automated Extraction Pipeline for YOLO Training Data

Paul-Catalin Ochis

Master's in Artificial Intelligence & Distributed Computing
West University of Timișoara

Supervisor: Prof. Alin Brândușescu

January 2026

Outline

Project Overview

Objective

Create an automated pipeline to generate YOLO-ready training datasets from raw automotive component images. The goal is to minimize manual annotation effort while maximizing dataset quality for deep learning applications.

Challenge

Manual annotation of 275 source images containing multiple overlapping components would require significant time investment and expertise. Each image may contain 1-10 distinct automotive parts that need precise

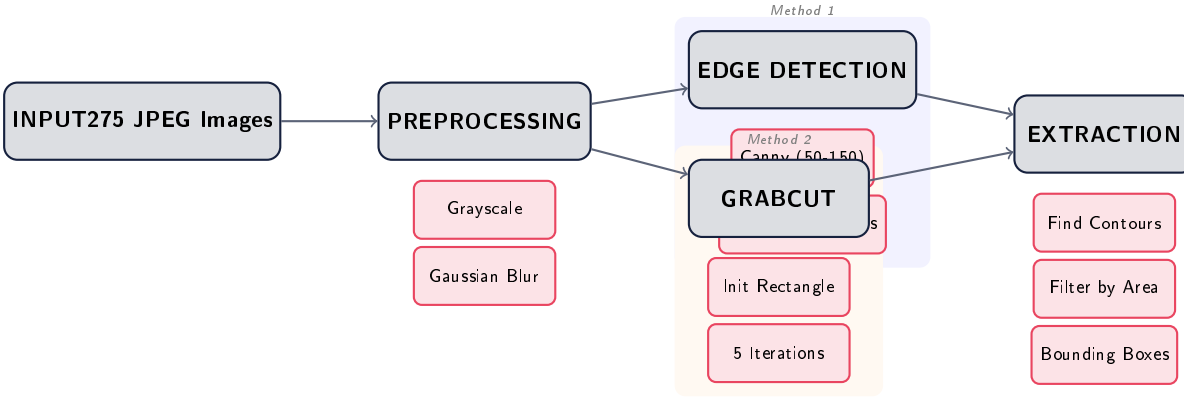
Solution

Automated component extraction pipeline combining classical computer vision techniques:

- **Canny Edge Detection** – identifies component boundaries
- **GrabCut Segmentation** – separates foreground from background
- **Morphological Operations** – refines detection masks
- **Contour Analysis** – extracts bounding boxes

Key Results

System Architecture Overview



Dual-Method Approach: The system implements two parallel extraction methods. Edge detection excels at high-contrast

Output Generation: Each detected component produces a cropped image, YOLO-format label file, and entry in the tracking manifest for

Pipeline Implementation Details

Edge Detection Pipeline

The Canny edge detector identifies intensity gradients that correspond to component boundaries:

```
gray = cv2.cvtColor(img, BGR2GRAY)
gray = cv2.GaussianBlur(gray, (5,5), 0)
edges = cv2.Canny(gray, 50, 150)
```

Morphological operations clean up the edge map:

```
kernel = cv2.getStructuringElement(
    MORPH_RECT, (9,9))
edges = cv2.dilate(edges, kernel,
iter=2)
edges = cv2.morphologyEx(edges,
    MORPH_CLOSE, kernel, iter=2)
```

GrabCut Segmentation Pipeline

GrabCut uses iterative graph-cut optimization to separate foreground objects from background:

```
mask = np.zeros(img.shape[:2], uint8)
rect = (w*0.05, h*0.05, w*0.9, h*0.9)
cv2.grabCut(img, mask, rect,
    bgdModel, fgdModel, 5,
    GC_INIT_WITH_RECT)
```

Extract foreground mask:

```
mask2 = np.where(
    (mask == GC_FGD)|
    (mask == GC_PR_FGD), 1, 0)
```

Detection Parameters & Configuration

Key Parameters

Parameter	Value	Purpose
MIN_AREA_RATIO	1.2%	Filter noise
MAX_AREA_RATIO	75%	Exclude full-image
PADDING	25 px	Context margin
MAX_COMPONENTS	10	Per-image limit
CANNY_LOW	50	Edge sensitivity
CANNY_HIGH	150	Edge threshold
MORPH_KERNEL	9×9	Cleanup size
GRABCUT_ITERS	5	Segmentation passes

Target Component Classes

Five automotive component categories defined for future YOLO training:

- `brake_caliper`
- `alternator`
- `battery`
- `oil_filter`
- `air_filter`

Currently extracting as single class; manual labeling will assign specific categories.

Filtering Logic

Components must satisfy ALL criteria:

YOLO Label Format

```
class_id x_center y_center w h
```

Dataset Output Structure

Project Directory Layout

```
CV_Project/
+- Component_Images/          (275 source)
+- datasets/
|   +- all_components/
|       +- crops/             (1000+ JPG)
|       +- debug/             (visualizations)
|       +- manifests/         (CSV)
+- outputs/
|   +- all_edges/
|   |   +- crops/, labels/, images/
|   +- all_grabcut/
|       +- crops/, labels/, images/
+- src/                       (Python scripts)
+- configs/                   (YAML config)
```

CSV Manifest Structure

Complete tracking of all extracted components:

- `crop_filename` – Output file name
- `crop_path` – Absolute path to crop
- `source_image` – Original image path
- `img_id` – Source image identifier (1-275)
- `component_idx` – Component number (1-10)
- `x1, y1, x2, y2` – Pixel coordinates
- `width, height` – Crop dimensions

Enables full traceability from any crop back to its source location in the original image.

Results & Statistics

Images Processed	Components Extracted	Success Rate
248	1000+	90.2%
out of 275 images	individual crops	pipeline efficiency

Component Distribution Analysis

Distribution of components per successfully processed image:

- **1–2 components:** ∼35% of images (simple scenes)
- **3–5 components:** ∼45% of images (typical case)
- **6–10 components:** ∼20% of images (complex scenes)

Failed Images Analysis (27 total)

IDs: 34, 35, 150, 154, 165, 166, 170, 179, 185, 187, 193–195, 197, 210–211, 225–226, 230, 239, 244–245, 255–258, 272

Primary Causes:

- Low contrast between component and background
- Unusual lighting conditions
- Highly textured backgrounds

Conclusion & Future Work

Project Achievements

- ✓ **Automated Pipeline** – End-to-end extraction from raw images to YOLO-ready dataset
- ✓ **Dual Methods** – Edge detection and GrabCut segmentation for robust extraction
- ✓ **High Success Rate** – 90.2% of images successfully processed
- ✓ **Complete Traceability** – CSV manifests link every crop to source
- ✓ **Debug Outputs** – Visual verification of all detections

Future Development Roadmap

1. **Manual Review** – Quality check and correction of extracted crops
2. **Class Labeling** – Assign specific component categories to each crop
3. **YOLOv11 Training** – Train object detection model on curated dataset
4. **Multi-class Expansion** – Extend to all 5 component classes
5. **Real-time Application** – Deploy detection model for live use
6. **Performance Benchmarking** – Evaluate mAP, precision, recall metrics

Thank You!

Questions & Discussion

`github.com/Ochis27/CV_Car_Component_Dataset`

Python 3.x | OpenCV | NumPy | YOLO Format

Paul-Catalin Ochis — West University of Timișoara
Computer Vision — January 2026