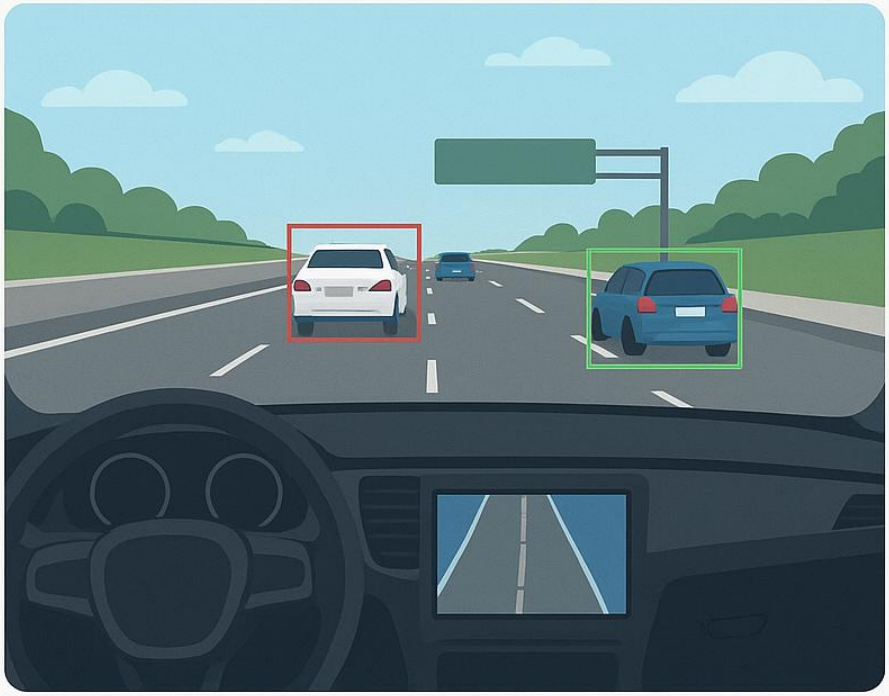
AUTONOMOUS DRIVING DATA LABELING PROJECT



MOHAMED RAAIZ

Autonomous Driving Data Labeling Project

Core Introduction

Autonomous driving relies heavily on **computer vision models** to recognize and respond to the environment. A key step is **data labeling**, where images or video frames from vehicle cameras are annotated with **objects, lanes, traffic signs, and pedestrians**. Accurate labeling ensures the AI can make **safe driving decisions** in real time.

Key Points

1. **Object Identification:**
    - Detect cars, trucks, bikes, pedestrians, traffic signs, and traffic lights.
2. **Bounding Boxes / Segmentation:**
    - Draw precise bounding boxes or pixel-level masks around objects.
3. **Lane and Road Markings:**
    - Annotate lanes, crosswalks, and road boundaries.
4. **Occlusion & Distance Estimation:**
    - Note partially visible objects and approximate distance if possible.
5. **Consistency & Accuracy:**
    - Follow strict guidelines to ensure AI models learn correctly.
6. **Safety Implications:**
    - Correct labeling is critical; mistakes can lead to unsafe driving behavior.

Coding

In This I Used A Dummy Data Set As Sample

```
# Install LabelImg for image annotation
!pip install labelImg
```

```
Collecting labelImg
  Downloading labelImg-1.8.6.tar.gz (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.7/247.7 kB 5.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting pyqt5 (from labelImg)
  Downloading PyQt5-5.15.11-cp38-abi3-manylinux_2_17_x86_64.whl.metadata (2.1 kB)
Requirement already satisfied: lxml in /usr/local/lib/python3.12/dist-packages (from labelImg) (5.4.0)
Collecting PyQt5-sip<13,>=12.15 (from pyqt5->labelImg)
  Downloading PyQt5_sip-12.17.0-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_64.whl.metadata (472 bytes)
Collecting PyQt5-Qt5<5.16.0,>=5.15.2 (from pyqt5->labelImg)
  Downloading PyQt5_Qt5-5.15.17-py3-none-manylinux2014_x86_64.whl.metadata (536 bytes)
Downloading PyQt5-5.15.11-cp38-abi3-manylinux_2_17_x86_64.whl (8.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 108.3 MB/s eta 0:00:00
Downloading PyQt5_Qt5-5.15.17-py3-none-manylinux2014_x86_64.whl (61.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.1/61.1 MB 16.9 MB/s eta 0:00:00
Downloading PyQt5_sip-12.17.0-cp312-cp312-manylinux_2_5_x86_64.manylinux1_x86_64.whl (281 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.9/281.9 kB 23.3 MB/s eta 0:00:00
Building wheels for collected packages: labelImg
  Building wheel for labelImg (setup.py) ... done
  Created wheel for labelImg: filename=labelImg-1.8.6-py2.py3-none-any.whl size=261521 sha256=acbae42569b4fb38f6e2a1a1c840e8f9933147e8153a41ed9e36ce945c48ab52
  Stored in directory: /root/.cache/pip/wheels/e6/1c/26/d1b603062180ce80890567fb7e1e837db08bdea938aaea77f3
Successfully built labelImg
Installing collected packages: PyQt5-Qt5, PyQt5-sip, pyqt5, labelImg
Successfully installed PyQt5-Qt5-5.15.17 PyQt5-sip-12.17.0 labelImg-1.8.6 pyqt5-5.15.11
```

```
[2] # Launch LabelImg GUI
    !labelImg
```

```
qt.qpa.plugin: Could not load the Qt platform plugin "xcb" in "" even though it was found.
This application failed to start because no Qt platform plugin could be initialized. Reinstalling the application may fix this problem.

Available platform plugins are: eglfs, linuxfb, minimal, minimalegl, offscreen, vnc, wayland-egl, wayland, wayland-xcomposite-egl, wayland-xcomposite-glx, webgl, xcb.
```

Mohamed Raaiz

```python
import os
import xml.etree.ElementTree as ET
import pandas as pd

# Step 1: Create folders
os.makedirs("images", exist_ok=True)
os.makedirs("annotations", exist_ok=True)

# Step 2: Generate dummy image filenames
image_filenames = [f"image_{i}.jpg" for i in range(1, 6)]  # 5 images

# Step 3: Generate dummy XML annotation files
for img_name in image_filenames:
    annotation = ET.Element("annotation")

    filename = ET.SubElement(annotation, "filename")
    filename.text = img_name

    # Add 1-3 random objects per image
    objects = [
        {"name": "car", "bbox": [50, 50, 200, 200]},
        {"name": "pedestrian", "bbox": [150, 100, 180, 220]},
        {"name": "traffic_light", "bbox": [300, 50, 320, 100]}
    ]

    for obj in objects:
        obj_elem = ET.SubElement(annotation, "object")
        name_elem = ET.SubElement(obj_elem, "name")
        name_elem.text = obj["name"]

        bndbox = ET.SubElement(obj_elem, "bndbox")
        xmin = ET.SubElement(bndbox, "xmin")
        xmin.text = str(obj["bbox"][0])
        ymin = ET.SubElement(bndbox, "ymin")
        ymin.text = str(obj["bbox"][1])
        xmax = ET.SubElement(bndbox, "xmax")
        xmax.text = str(obj["bbox"][2])
        ymax = ET.SubElement(bndbox, "ymax")
        ymax.text = str(obj["bbox"][3])
```

```python
    tree = ET.ElementTree(annotation)
    xml_path = os.path.join("annotations", img_name.replace(".jpg", ".xml"))
    tree.write(xml_path)

print("Dummy dataset and XML annotations created successfully!")
```

```
Dummy dataset and XML annotations created successfully!
```

```python
# Step 2: Read annotations
annotation_folder = "annotations/"

xml_files = [f for f in os.listdir(annotation_folder) if f.endswith(".xml")]

annotations = []
for xml_file in xml_files:
    tree = ET.parse(os.path.join(annotation_folder, xml_file))
    root = tree.getroot()

    image_name = root.find("filename").text
    for obj in root.findall("object"):
        label = obj.find("name").text
        bbox = obj.find("bndbox")
        xmin = int(bbox.find("xmin").text)
        ymin = int(bbox.find("ymin").text)
        xmax = int(bbox.find("xmax").text)
        ymax = int(bbox.find("ymax").text)
        annotations.append([image_name, label, xmin, ymin, xmax, ymax])

df = pd.DataFrame(annotations, columns=["Image", "Label", "Xmin", "Ymin", "Xmax", "Ymax"])
df.head()
```

	Image	Label	Xmin	Ymin	Xmax	Ymax
0	image_1.jpg	car	50	50	200	200
1	image_1.jpg	pedestrian	150	100	180	220
2	image_1.jpg	traffic_light	300	50	320	100
3	image_3.jpg	car	50	50	200	200
4	image_3.jpg	pedestrian	150	100	180	220

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.to_csv("dummy_autonomous_driving_annotations.csv", index=False)
print("Saved dummy annotations to dummy_autonomous_driving_annotations.csv")
```

```
Saved dummy annotations to dummy_autonomous_driving_annotations.csv
```

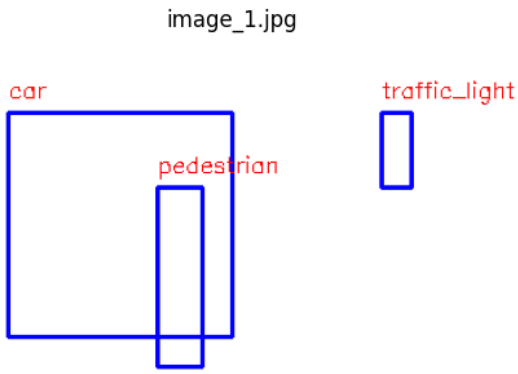
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Visualize annotations for the first image
image_name = df.iloc[0]["Image"]

# Create a blank dummy image
img = np.zeros((400, 400, 3), dtype=np.uint8) + 255  # white canvas

# Draw bounding boxes
for _, row in df[df["Image"] == image_name].iterrows():
    cv2.rectangle(img, (row["Xmin"], row["Ymin"]), (row["Xmax"], row["Ymax"]), (255,0,0), 2)
    cv2.putText(img, row["Label"], (row["Xmin"], row["Ymin"]-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 1)

plt.figure(figsize=(6,6))
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title(image_name)
plt.axis("off")
plt.show()
```



Conclusion

Autonomous driving data labeling ensures **AI systems understand the vehicle's environment**. Labeled data improves:

- **Collision avoidance** through accurate object detection.
- **Traffic rule compliance** via traffic sign recognition.
- **Lane-keeping and navigation** by understanding lane markings and obstacles.

Tip: Use advanced labeling tools like **CVAT, LabelImg, or Label Studio**, and always validate your annotations.

Advanced Python Code for Object Detection Labeling

Here's a **Python workflow** to annotate images for autonomous driving using Labeling and pandas for metadata storage. This example focuses on **bounding box annotations**.