```
from google.colab import files
uploaded = files.upload()
print("Uploaded:", list(uploaded.keys()))
```

```
Choose Files No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Accident.zip to Accident (1).zip
```

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
from IPython.display import display, HTML
import imutils

def detect_accident(img_bgr, resize_to=(640,480), params=None):
    if params is None:
        params = {
            "edge_density_thresh": 0.06,
            "area_ratio_thresh": 0.04,
            "red_ratio_thresh": 0.01
        }
    img = img_bgr.copy()
    img = cv2.resize(img, resize_to)
    h, w = img.shape[:2]
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (5,5), 0)
    edges = cv2.Canny(blurred, 50, 150)
    edge_density = edges.sum() / (255.0 * h * w)
    _, thresh = cv2.threshold(blurred, 120, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    total_area = sum(cv2.contourArea(c) for c in contours)
    area_ratio = total_area / (h * w)
    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower1 = np.array([0, 70, 50]); upper1 = np.array([10, 255, 255])
    lower2 = np.array([170,70,50]); upper2 = np.array([180,255,255])
    mask1 = cv2.inRange(hsv, lower1, upper1)
    mask2 = cv2.inRange(hsv, lower2, upper2)
    red_mask = cv2.bitwise_or(mask1, mask2)
    red_ratio = red_mask.sum() / (255.0 * h * w)

    flag = (edge_density > params["edge_density_thresh"]) or (area_ratio > params["area_ratio_thresh"])
    label = "ACCIDENT-LIKELY" if flag else "NO-ACCIDENT-LIKELY"
    reasons = []
    if edge_density > params["edge_density_thresh"]:
        reasons.append(f"edge_density={edge_density:.3f}")
    if area_ratio > params["area_ratio_thresh"]:
        reasons.append(f"area_ratio={area_ratio:.3f}")
    if red_ratio > params["red_ratio_thresh"]:
        reasons.append(f"red_ratio={red_ratio:.3f}")
    overlay = img.copy()
    cv2.drawContours(overlay, contours, -1, (0,255,0), 2)
    edges_color = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)
    edges_small = cv2.resize(edges_color, (int(w*0.25), h))
    overlay[:, -edges_small.shape[1]:] = edges_small

    return {
        "label": label,
        "edge_density": float(edge_density),
        "area_ratio": float(area_ratio),
        "red_ratio": float(red_ratio),
        "reasons": reasons,
        "overlay": overlay
    }

def show_image(img_bgr, title=None, figsize=(8,5)):
    img_rgb = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=figsize)
    plt.imshow(img_rgb)
    if title:
        plt.title(title)
    plt.axis("off")
    plt.show()
```

```
from pathlib import Path
out_dir = Path("/content/Accident_images")
img_paths = list(out_dir.rglob("*"))
img_paths = [p for p in img_paths if p.suffix.lower() in (".jpg",".jpeg",".png",".bmp",".tiff")]
print("Total images found:", len(img_paths))
```

```
N = 12
results = []
for p in img_paths[:N]:
    img = cv2.imread(str(p))
    if img is None:
        continue
    info = detect_accident(img)
    results.append({
        "file": str(p),
        "label": info["label"],
        "edge_density": info["edge_density"],
        "area_ratio": info["area_ratio"],
        "red_ratio": info["red_ratio"],
        "reasons": info["reasons"]
    })
    show_image(info["overlay"], title=f"{p.name} -> {info['label']}")

import pandas as pd
df = pd.DataFrame(results)
if not df.empty:
    display(df)
else:
    print("No images processed.")
```

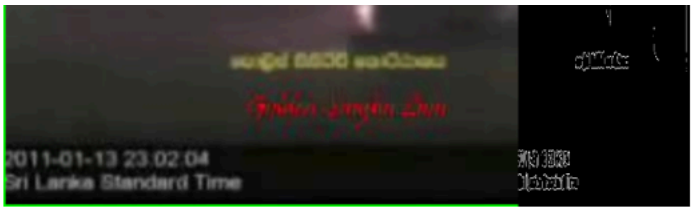
```
Total images found: 6191
```

5424.jpg -> ACCIDENT-LIKELY

5165.jpg -> ACCIDENT-LIKELY

1278.jpg -> ACCIDENT-LIKELY

3190.jpg -> ACCIDENT-LIKELY

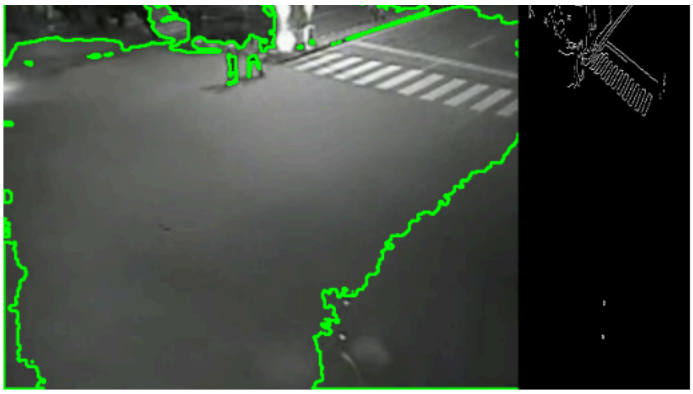


3759.jpg -> ACCIDENT-LIKELY

986.jpg -> ACCIDENT-LIKELY

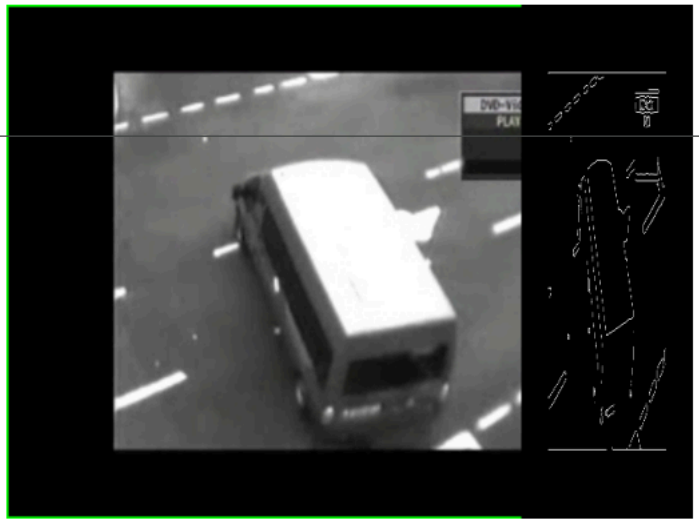
798.jpg -> ACCIDENT-LIKELY

2923.jpg -> ACCIDENT-LIKELY



1504.jpg -> ACCIDENT-LIKELY

241.jpg -> ACCIDENT-LIKELY



2037.jpg -> ACCIDENT-LIKELY

```
from pathlib import Path
p = Path("/content/Accident_images/1.jpg")
if p.exists():
    img = cv2.imread(str(p))
    info = detect_accident(img)
    print("Label:", info["label"])
    print("Reas:", info["reasons"])
    show_image(info["overlay"], title=p.name)
else:
    print("Change the path to an existing image. List images in /content/Accident_images")
```

```
Change the path to an existing image. List images in /content/Accident_images
```

```
import zipfile, os, shutil
from pathlib import Path

WORK = Path("/content")
OUT = WORK / "Accident_images"
OUT.mkdir(parents=True, exist_ok=True)

zips = sorted([p for p in WORK.iterdir() if p.suffix.lower() == ".zip"], key=lambda p: p.stat().st_size, reverse=True)
print("Found zip files (largest first):")
for p in zips:
    print(f" - {p.name} ({p.stat().st_size/1e6:.1f} MB)")

def try_unzip(zip_path, dest):
    try:
        with zipfile.ZipFile(zip_path, "r") as z:
            z.testzip()
            z.extractall(dest)
        return True, None
    except zipfile.BadZipFile as e:
        return False, "BadZipFile"
    except Exception as e:
        return False, str(e)

shutil.rmtree(OUT)
OUT.mkdir(parents=True, exist_ok=True)

unzipped = False
errors = []
for z in zips:
    ok, err = try_unzip(z, OUT)
    if ok:
        print(f"✅ Unzipped {z.name} -> {OUT}")
        unzipped = True
        break
    else:
        print(f"❌ Could not unzip {z.name}: {err}")
        errors.append((z.name, err))

if (not unzipped) and (len(zips) == 0):
    print("No .zip files found in /content. Please upload Accident.zip via the file upload cell or mount Drive.")
elif not unzipped:
    print("All candidate zips failed. Consider re-uploading a clean zip file or using the upload cell below.")
```

```
Found zip files (largest first):
 - Accident (1).zip (159.8 MB)
 - Accident.zip (159.8 MB)
✅ Unzipped Accident (1).zip -> /content/Accident_images
```

```
from pathlib import Path
IMG_EXT = {".jpg", ".jpeg", ".png", ".bmp", ".tiff"}
IMG_DIR = Path("/content/Accident_images")
image_paths = sorted([p for p in IMG_DIR.rglob("*") if p.suffix.lower() in IMG_EXT])

print("Images found:", len(image_paths))
if len(image_paths) > 0:
    for p in image_paths[:10]:
        print(" -", p.relative_to(IMG_DIR))
else:
    print("No images found. If you uploaded a nested zip, ensure images are inside Accident_images or re-upload.")
```

```
Images found: 6191
 - Accident/1.jpg
 - Accident/10.jpg
 - Accident/100.jpg
 - Accident/1000.jpg
```

```
 - Accident/1001.jpg
 - Accident/1002.jpg
 - Accident/1003.jpg
 - Accident/1004.jpg
 - Accident/1005.jpg
 - Accident/1006.jpg
```

```
import cv2, numpy as np
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
import imutils

def detect_accident_bgr(img_bgr, params=None, resize_to=(640,480)):
    """
    Input: BGR OpenCV image
    Returns: dict with label, metrics, reasons, and overlay image
    """
    if params is None:
        params = {
            "edge_thresh": 0.055,
            "area_ratio_thresh": 0.035,
            "red_ratio_thresh": 0.01,
            "combine_vote": 2
        }
    img = imutils.resize(img_bgr, width=resize_to[0]) if img_bgr.shape[1] != resize_to[0] else img_bgr.copy()
    h, w = img.shape[:2]
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    edges = cv2.Canny(blur, 50, 150)
    edge_density = float(edges.sum()) / (255.0 * h * w)

    _, thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
    contours, _ = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    total_area = sum(float(cv2.contourArea(c)) for c in contours)
    area_ratio = total_area / float(h * w)

    hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    lower1 = np.array([0, 70, 50]); upper1 = np.array([10, 255, 255])
    lower2 = np.array([170, 70, 50]); upper2 = np.array([180, 255, 255])
    m1 = cv2.inRange(hsv, lower1, upper1)
    m2 = cv2.inRange(hsv, lower2, upper2)
    red_mask = cv2.bitwise_or(m1, m2)
    red_ratio = float(red_mask.sum()) / (255.0 * h * w)

    votes = 0
    reasons = []
    if edge_density > params["edge_thresh"]:
        votes += 1
        reasons.append(f"edge_density={edge_density:.3f}")
    if area_ratio > params["area_ratio_thresh"]:
        votes += 1
        reasons.append(f"area_ratio={area_ratio:.3f}")
    if red_ratio > params["red_ratio_thresh"]:
        votes += 1
        reasons.append(f"red_ratio={red_ratio:.3f}")

    label = "ACCIDENT-LIKELY" if votes >= params["combine_vote"] else "NO-ACCIDENT-LIKELY"

    overlay = img.copy()
    drawn = 0
    for c in contours:
        if cv2.contourArea(c) < 50:
            continue
        cv2.drawContours(overlay, [c], -1, (0,255,0), 2)
        drawn += 1
        if drawn >= 30:
            break
    edges_color = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR)
    strip_w = int(w * 0.22)
    edges_small = cv2.resize(edges_color, (strip_w, h))
    overlay[:, -strip_w:] = edges_small
    color = (0,0,255) if label.startswith("ACCIDENT") else (0,255,0)
    cv2.putText(overlay, f"{label}", (10,30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, color, 2)
    cv2.putText(overlay, f"E:{edge_density:.3f} A:{area_ratio:.3f} R:{red_ratio:.3f}", (10,60), cv2.FONT_HERSHEY_SIMPLEX, 0

    return {
        "label": label,
        "edge_density": edge_density,
        "area_ratio": area_ratio,
        "red_ratio": red_ratio,
        "reasons": reasons,
        "overlay": overlay
```

```
    }

def show_bgr(img_bgr, title=None, figsize=(10,6)):
    if title:
        print(title)
    cv2_imshow(img_bgr)
```

```
import pandas as pd
from tqdm import tqdm
import cv2, os

if len(image_paths) == 0:
    print("No images found to process. Please unzip/upload images.")
else:
    rows = []
    params = None
    for p in tqdm(image_paths, desc="Processing images"):
        try:
            img = cv2.imread(str(p))
            if img is None:
                rows.append({"file": str(p.relative_to(IMG_DIR)), "error": "read-failed"})
                continue
            info = detect_accident_bgr(img, params=params)
            rows.append({
                "file": str(p.relative_to(IMG_DIR)),
                "label": info["label"],
                "edge_density": info["edge_density"],
                "area_ratio": info["area_ratio"],
                "red_ratio": info["red_ratio"],
                "reasons": "; ".join(info["reasons"])
            })
        except Exception as e:
            rows.append({"file": str(p.relative_to(IMG_DIR)), "error": str(e)})

    df = pd.DataFrame(rows)
    out_csv = "/content/predictions.csv"
    df.to_csv(out_csv, index=False)
    print("Saved predictions to", out_csv)
    print(df['label'].value_counts(dropna=True))
```

```
Processing images: 100%|██████████| 6191/6191 [01:08<00:00, 90.54it/s]
Saved predictions to /content/predictions.csv
label
ACCIDENT-LIKELY       3672
NO-ACCIDENT-LIKELY    2519
Name: count, dtype: int64
```

```
import random, cv2
acc = df[df['label']=="ACCIDENT-LIKELY"]
noacc = df[df['label']=="NO-ACCIDENT-LIKELY"]

def show_sample_rows(rows, title_prefix=""):
    for _, r in rows.iterrows():
        p = IMG_DIR / r['file']
        img = cv2.imread(str(p))
        info = detect_accident_bgr(img)
        show_bgr(info['overlay'], title=f"{title_prefix}{r['file']} -> {r['label']}")

print("Showing up to 6 ACCIDENT-LIKELY examples (if any):")
if len(acc)>0:
    show_sample_rows(acc.head(6))
else:
    print("No ACCIDENT-LIKELY examples found.")

print("Showing up to 6 NO-ACCIDENT examples (random):")
if len(noacc)>0:
    sample_no = noacc.sample(min(6, len(noacc)))
    show_sample_rows(sample_no)
else:
    print("No NO-ACCIDENT examples found.")
```