Actualizare: Detectarea Semnului *WARNING*

1 Scopul actualizării

Versiunea anterioară detecta doar semnele *YIELD*, *STOP*, *NO ENTRY* şi *SPEED*. Obiectivul a fost extinderea detectorului cu semnul triangular de avertizare *WARNING* şi corectarea logicii de determinare a orientării triunghiurilor.

2 Modificări în pipeline

1. **Îndreptarea conturului**
   După extragerea conturului original c, se aplică:
   `convexHull(c, hull);`
   pentru a elimina muchiile curbate introduse de colţurile rotunjite.

2. **Aproximare poligonală mai permisivă**
   `approxPolyDP(hull, approx, 0.04 * peri, true);`
   unde `peri = arcLength(hull, true)`.
   Creşterea lui ε la 4% din perimetru compactează colţurile şi produce exact trei vârfuri pentru semnul triangular.

3. **Clasificare nouă *WARNING***

```
// semn WARNING: triunghi cu apex in sus + chenar rosu subire
else if (approx.size() == 3 &&
        isTrianglePointingUp(approx) &&
        redRatio > 0.15 && redRatio < 0.60) {
    label = "WARNING";
    color = Scalar(0, 200, 255); // portocaliu
    matched = true;
}
```

4. **Orientare triunghi – funcţii rescrise**
   Determinarea apex-ului se face statistic: un singur vârf trebuie să fie de o parte a centrului de greutate, celelalte două de partea opusă.

3 Noile funcţii de orientare

```cpp
bool isTrianglePointingUp(const vector<Point>& tri) {
    Moments m = moments(tri);
    Point2f c(m.m10 / m.m00, m.m01 / m.m00);
    int above = 0, below = 0;
    for (auto& p : tri) (p.y < c.y) ? ++above : ++below;
    return above == 1 && below == 2; // apex sus
}

bool isTrianglePointingDown(const vector<Point>& tri) {
    Moments m = moments(tri);
    Point2f c(m.m10 / m.m00, m.m01 / m.m00);
    int above = 0, below = 0;
    for (auto& p : tri) (p.y < c.y) ? ++above : ++below;
    return below == 1 && above == 2; // apex jos
}
```

4 Rezultat exemplar

Figura 1: Semnul *WARNING* detectat şi etichetat corect.