Shahed-136 Drone Detection and Localization System

CyberDefense Hackathon - Support for Ukraine

Team: Drone Detective (4 members)
Duration: Friday 6pm → Sunday noon (42 hours)
Objective: Develop a low-cost kamikaze drone detection solution

---

a) What problem are we solving?

Operational context

Since 2022, Ukraine has faced massive attacks from Iranian kamikaze drones Shahed-136 (rebranded Geran-2 by Russia). These drones, used by the hundreds against civilian and military infrastructure, pose several challenges:

Identified problems:

- **Difficult detection:** Flies at low altitude (50-300m), moderate speed (185 km/h)
- **Asymmetric cost:** A Shahed costs around 20,000 euros, intercepting them with Patriot missiles costs 4 million euros
- **Defense saturation:** Launched in swarms of dozens of units simultaneously
- **Lack of civilian detection means:** Military radars do not cover the entire territory

Our solution

We propose a visual detection system using artificial intelligence that transforms any surveillance camera into a drone detector. The system provides:

1. **Automatic detection** of Shahed-136 with 99.5% accuracy
2. **Real-time distance estimation** (without additional sensor)
3. **Continuous tracking** to follow trajectory
4. **Early alerts** to activate defenses or evacuate

Key advantage: Deployable within hours on existing camera networks (urban video surveillance, security cameras, etc.)

---

b) How are we solving the problem technically?

System architecture

Our solution is based on three main components:

1. Deep learning detection (YOLOv11)

Why YOLOv11?

- Speed: 30+ FPS on modest hardware
- Accuracy: Recent architecture (2024), optimized for small objects
- Efficiency: "Nano" model (2.5M parameters) = low consumption

Our training approach:

```
Dataset found: "Military-Units-24v07v2023"
  → 930 training images
  → 1 class: "military-drone"

Training: 100 epochs with early stopping
  → GPU: 2h computation
  → Results: mAP50 = 99.5% (near perfect)
```

Anti-false-positive filters: To avoid detecting birds, civilian aircraft or other objects, we apply:

- Bbox size filtering (800-400,000 pixels²)
- Characteristic length/width ratio (0.8-4.5)
- High confidence threshold (55%)

2. 3D distance estimation (without sensor)

The challenge: How to measure distance without LiDAR or radar?

Our geometric solution:

We know the real dimensions of the Shahed-136:

- Length: 3.5 m
- Wingspan: 2.5 m

By measuring the apparent size of the drone in the image (in pixels), we can find its distance by triangulation:

```
Distance = (Real_size × Camera_focal) / Image_size
```

Improvements made:

- **Automatic view angle detection** (lateral/frontal/oblique)
- **Weighted multi-dimension calculation** (width + height + diagonal)
- **Temporal smoothing** over 20 frames (sliding median)
- **Outlier rejection** (outliers > 2.5× median deviation)

Result: stable distances with coefficient of variation < 15%

3. Smart tracking

Problem encountered: The YOLO model detects frame by frame, without "memory". If the drone is temporarily hidden (cloud, building), the counter goes wild.

Our tracker:

- **Temporal memory:** keeps last position for 5 frames
- **IoU matching:** verifies it's the same object (30% threshold)

- **Unique counter:** counts once per drone, even if temporarily lost

Benefit: Continuous tracking even with temporary occlusions

Technical stack

```
Language: Python 3.12
AI Framework: Ultralytics YOLOv11 (PyTorch)
Vision: OpenCV 4.x
Scientific computing: NumPy

Hardware tested:
  - Laptop AMD Ryzen 7 5825U (CPU)
  - Result: ~5 FPS in debug mode
  - Deployment target: GPU or optimized CPU → 30 FPS
```

---

c) How will this be deployed and mass-manufactured?

Government deployment scenario

Phase 1: Existing infrastructure (0-3 months)

Leverage already installed video surveillance network:

- Municipal cameras (intersections, public squares)
- Critical infrastructure security cameras (power plants, bridges, dams)
- Volunteer private cameras (businesses, individuals)

Cost per surveillance point:

```
Scenario A (existing camera reuse):
  - Camera: 0 euros (already installed)
  - Mini-PC (Intel NUC, Raspberry Pi 5): 200-500 euros
  - Software installation: 1h/site
  → TOTAL: 200-500 euros per point

Scenario B (new installation):
  - Outdoor IP camera: 150 euros
  - Mini-PC: 300 euros
  - Installation: 200 euros
  → TOTAL: 650 euros per point
```

Comparison with conventional solutions:

- Mobile military radar: 500,000 - 2,000,000 euros
- Commercial anti-drone system: 50,000 - 200,000 euros
- **Our solution: < 1,000 euros** (50-2000x cheaper factor)

Network architecture

```
[Local cameras] → [Edge processing (mini-PC)] → [Central server]
                                                        ↓
                                                [Dashboard]
                                                [SMS/App alerts]
                                                [Forces interface]
```

Advantages of this architecture:

- **Local processing:** no need for high bandwidth (only alerts)
- **Resilience:** if network fails, each camera continues detecting
- **Scalability:** adding cameras without central server overload

Mass production

Step 1: Standardized kit

```
"SentinelDrone v1.0" kit contents:
  - Pre-configured mini-PC (bootable system image)
  - Compatible IP camera (optional)
  - Cables and mounting hardware
  - 1-page installation guide

Installation time: 30 minutes per site
Operator training: 1 hour
```

Step 2: Local manufacturing

Components available worldwide:

- Raspberry Pi, Intel NUC: global production
- IP cameras: Hikvision, Dahua, Western brands
- Software: open-source, infinitely duplicable

Production capacity:

```
Target 1: 100 units/month (medium city coverage)
Target 2: 1,000 units/month (complete region)
Target 3: 10,000 units/month (entire country)
```

Constraint: mini-PC supply (current global shortage)

Integration with existing systems

Compatible with:

- Military command centers (JSON/XML data export)
- Civil alert systems (SMS, sirens)
- Anti-aircraft defense networks (drone GPS coordinates)
- Mobile applications (citizen notifications)

---

d) What have we achieved during the hackathon?

Technical achievements (42h)

Friday evening (6pm-midnight): Research & Dataset

- Done: Shahed-136 drone problem identification
- Done: Dataset research (3h prospecting)
- Done: "Military-Units-24v07v2023" discovery (930 images)
- Done: Quality verification: correct annotations, angle diversity

Saturday (9am-midnight): Training & Detection

- Done: Environment setup (YOLOv11, dependencies)
- Done: Dataset preparation (data.yaml correction)
- Done: Model training: 99.5% mAP50 achieved!
- Done: Real-time detection implementation
- Issue discovered: overly sensitive detection (false positives)
- Done: Added geometric filters, problem solved

Saturday night (midnight-3am): Distance measurement

- Done: Shahed-136 dimension research (Wikipedia, OSMP sources)
- Done: Geometric calculation implementation
- Issue: inconsistent distances (astronomical values)
- Done: Debug - PnP unstable, disabled it, focused on geometric method

Sunday morning (9am-noon): Finalization & Robustness

- Done: Tracking system (IoU matching, 5-frame memory)
- Done: Distance smoothing (20-frame sliding median)
- Done: Anti-outlier filter (rejection > 2.5× median)
- Done: Improved weighted multi-dimension calculation
- Done: User interface (HUD with real-time stats)
- Done: Final video tests

Final metrics

Detection:

- Precision (mAP50): **99.5%**
- Precision (mAP50-95): **98.0%**
- Recall: **99.9%**
- Speed: 5 FPS (laptop CPU) → 30+ FPS expected (GPU)

Distance:

- Method: Multi-dimension geometric
- Stability: Coefficient of variation < 15%
- Confidence interval: Displayed in real-time (5-95% percentile)

Tracking:

- Unique counter: 1 drone = 1 count (no double counting)
- Memory: 5 frames without detection tolerated
- Matching: 30% IoU (robust to small movements)

Code produced

```
Project structure:
├── train_shahed_model.py       (YOLOv11 training)
├── detect_shahed_distance.py   (complete system)
├── shahed136_detector.pt       (trained model, 5 MB)
└── Military-Units-24v07v2023/  (dataset)
```

```
Lines of code: ~600 Python lines
Comments: French + inline documentation
Modularity: 3 classes (Tracker, Detection, Distance)
```

Challenges encountered and solutions

Problem	Impact	Solution found	Time lost
PnP gave aberrant distances	Blocking	Deactivation, focus on geometric	4h
False positives (walls, people)	Critical	Strict geometric filters	3h
Unstable tracking (crazy counter)	Major	IoU system + memory	2h
Variable distances by angle	Moderate	Weighted multi-dimension calculation	3h
Dataset introduction	Minor	Automatic data.yaml creation	1h

Total obstacles: 13h out of 42h, which is 31% of time

What remains to be done (post-hackathon)

Short term (week 1):

- GPU optimization (CUDA, TensorRT) to reach 60 FPS target
- Tests on Raspberry Pi 5 (target hardware validation)
- Real camera calibration (FOV, distortion)

Medium term (month 1):

- Web monitoring interface (real-time dashboard)
- REST API for third-party system integration
- Multi-class detection (Shahed-131, other drones)
- Training on expanded dataset (2000+ images)

Long term (3+ months):

- Night detection (infrared cameras)
- Trajectory prediction (AI + physics)
- Optimal interception calculation (defense guidance)
- Pilot deployment (10 test sites)

Conclusion

In 42 hours, we developed a functional anti-drone detection system achieving 99.5% accuracy for a cost below 1,000 euros per surveillance point.

Potential impact:

- **Economic:** 50-2000x cheaper factor than military radars
- **Speed:** Deployable within hours on existing infrastructure
- **Scalability:** Mass production possible with standard components
- **Defensive:** Protects civilian infrastructure without offensive weapons

This project demonstrates that with artificial intelligence and consumer hardware, it is possible to create accessible defense solutions, even for countries with limited military budgets.

Our conviction: Technology should serve to protect civilians, and open-source allows democratizing access to these vital tools.

Contacts:
Team: Drone Detective
mail : martins.andre.infor@gmail.com