RUNWAY INSPECTION FOR AIRPORTS USING AUTONOMOUS FIXED WING

Group V

Introduction

Foreign Object Detection on Runway Using Computer Vision

Foreign object debris (FOD) poses a significant threat to aviation safety. Even small objects like birds, tools, or metal fragments can cause serious damage to aircraft and endanger lives.

Improved Efficiency

- Manual inspections are labor-intensive, time-consuming, and prone to missing small or obscured objects.
- Computer vision automates the detection process, enabling continuous monitoring and faster identification of FOD.

Data Collection and Analysis

- Computer vision systems collect and analyze data on the type, frequency, and location of FOD.
- These insights can inform better preventive measures, enhancing overall runway safety.

Enhanced Accuracy

- Deep learning algorithms trained on extensive datasets of runway images can identify a broader range of FOD with greater precision than human inspectors.

24/7 Operation

- Unlike human inspectors, computer vision systems can operate continuously, ensuring constant vigilance across all weather conditions and times of day.

Reduced Costs

- Automating the FOD detection process reduces the dependence on manpower, leading to significant cost savings over time.

DATA COLLECTION: Building a Robust Dataset from Scratch

To ensure our AI model could effectively detect foreign objects on airport runways, we explored multiple strategies to gather and create a dataset tailored to real-world conditions.

APPROACH 1 : Sourcing Publicly Available Datasets

We began by searching for pre-existing datasets from a variety of trusted sources:

- Research papers and academic publications
- University research group repositories
- Public platforms like Roboflow, NASA's data portal, and other open-source communities

Challenge: Although these sources had datasets related to object detection and runways, none of them were comprehensive or specific enough to meet the requirements of our project. Most lacked diversity, quality, or relevance to real-world runway environments.

APPROACH 2 : Generating synthetic dataset using blender

Generating Synthetic Data Using Blender

Next, we tried creating a synthetic dataset by modeling runway scenarios in Blender. This allowed us to:

- Simulate different lighting and object conditions
- Control the placement and type of foreign objects
- Rapidly generate a large volume of training data

Challenge: While the synthetic dataset helped in bootstrapping model training, it lacked the realism of actual runway conditions. As a result, the model overfitted and failed to generalize well when tested on real-world images.

DATA COLLECTION: Building a Robust Dataset from Scratch

After facing limitations with public datasets and synthetic data, we shifted our focus to building a custom, real-world dataset. This hands-on approach allowed us to capture images that truly reflected the complexities of runway environments, forming the foundation for a model that performs reliably in real scenarios.

APPROACH 3 : Capturing and Augmenting Real-World Data

Recognizing the gap, we moved to a more hands-on approach:

- Captured high-resolution images of runway simulations across varied lighting and object conditions
- Labeled the images using Roboflow for precise object detection
- Used this labeled data as a base to generate semi-synthetic augmented images to enrich the dataset

Result: This hybrid approach combined the authenticity of real-world images with the scalability of synthetic augmentation, significantly improving our model's accuracy and robustness in real-world testing scenarios.

Real-World Dataset Samples : Below are sample images from our custom dataset, captured under diverse runway conditions. These images reflect the variety and realism needed to train our AI model effectively, forming the core of our data-driven solution.






Model Exploration & Selection

To determine the most effective object detection model for our AI-powered runway inspection system, we conducted an in-depth evaluation of several cutting-edge models. This process was informed by an extensive review of research papers, technical articles, and academic benchmarks, helping us shortlist models that are widely recognized for their performance in object detection tasks.

We focused our experiments on models such as YOLOv5, YOLOv8, Faster R-CNN, SSD, RetinaNet, and EfficientDet, each known for their unique balance of accuracy, speed, and complexity. Using our custom dataset, we trained and tested each model to assess how well it could detect foreign objects on runways under real-world conditions.

We tested each model on our custom dataset, focusing on key factors like accuracy, inference speed, real-world generalization, and ease of deployment. The results are summarized in the table below:

Model	Accuracy	Inference Speed	Real-World Generalization	Ease of Integration	Overall Verdict
YOLOv5	High	Fast	Good	Easy	👍 Reliable
Faster R-CNN	Very High	Slow	Excellent	Moderate	✅ Accurate, but slow
SSD	Moderate	Fast	Average	Easy	⚠️ Limited accuracy
RetinaNet	High	Moderate	Good	Moderate	✅ Balanced
EfficientDet-D1	High	Moderate	Good	Moderate	✅ Efficient
YOLOv8	Very High	Very Fast	Excellent	Very Easy	⭐ Best Overall
YOLO-CS	Very High	Very Fast	Excellent (on custom datasets)	Easy	⭐ Great for custom scenarios

YOLOv8 – The Chosen Model for Runway Inspection

YOLOv8 (You Only Look Once, version 8) is the latest and most advanced model in the YOLO family, developed by Ultralytics. It is designed for high-speed and high-accuracy object detection, making it ideal for real-time applications like runway surveillance and safety systems.

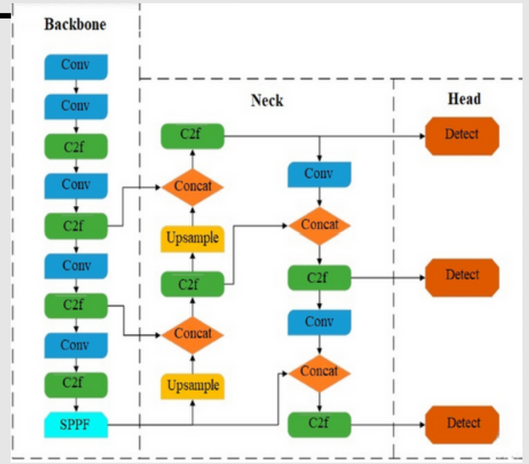
Key Features of YOLOv8:

- High Accuracy: Improved architecture and training techniques result in better precision, even in complex environments.
- Fast Inference: Optimized for real-time performance, achieving up to 55+ FPS on modern GPUs.
- Anchor-Free Design: Unlike previous YOLO versions, YOLOv8 uses an anchor-free detection approach, simplifying training and improving generalization.
- Lightweight and Scalable: Suitable for deployment on both edge devices and high-performance servers.
- Built-in AutoML Tools: Comes with support for tasks like object detection, instance segmentation, pose estimation, and classification.
- Easy to Use: Integrated with the Ultralytics Python library, making training, inference, and deployment highly user-friendly.

Why YOLOv8 for Our Project?

YOLOv8 struck the perfect balance between speed, accuracy, and ease of deployment. It handled our custom dataset with high precision, adapted well to real-world runway conditions, and seamlessly integrated into our pipeline – making it the most reliable model for our use case.

YOLO V8 Architecture :



FOD Detection Using AI

Custom YOLOv8-based object detection model trained on real-world runway images for accurate and real-time FOD detection.

```
# Clone the official Ultralytics YOLO repository
!git clone https://github.com/ultralytics/ultralytics.git

# Install the Ultralytics YOLO package
!pip install ultralytics

# Install necessary libraries for image processing and visualization
!pip install opencv-python numpy matplotlib tqdm

# Train YOLOv8 on custom dataset
# - task=detect indicates object detection
# - mode=train specifies training mode
# - model=yolov8n.pt loads the YOLOv8 nano pre-trained weights
# - data=data.yaml is the path to dataset configuration
# - epochs=50 sets the number of training epochs
# - imgsz=640 defines the input image size
!yolo task=detect mode=train model=yolov8n.pt data=data.yaml epochs=50 imgsz=640

# Validate the trained model using the validation set
# - mode=val indicates validation mode
# - model=best.pt loads the best weights saved during training
!yolo task=detect mode=val model=runs/detect/train/weights/best.pt data=data.yaml
```

```
# Run object detection on images inside the 'test_images' folder
# - mode=predict is for inference
# - model=best.pt uses the best trained model
# - source="test_images" is the folder containing input images
# - save=True saves the output images with bounding boxes
!yolo task=detect mode=predict model=./runs/detect/train/weights/best.pt source="test_images" save=True

# Import libraries for displaying images
import cv2
import matplotlib.pyplot as plt
import os
import math

# Get the list of output images generated by YOLO
output_dir = "runs/detect/predict"
output_images = [f for f in os.listdir(output_dir) if f.endswith(('.jpg', '.png'))]

# Dynamically set grid size based on number of output images
num_images = len(output_images)
cols = 4  # Number of columns in the display grid
rows = math.ceil(num_images / cols)  # Number of rows based on total images

# Create a grid plot to display images
fig, axes = plt.subplots(rows, cols, figsize=(20, 20))  # Set large figure size
axes = axes.flatten()
```

```
# Display each image with predicted bounding boxes
for i, img_name in enumerate(output_images):
    img = cv2.imread(f"{output_dir}/{img_name}")  # Read the image
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)    # Convert BGR to RGB for plotting

    axes[i].imshow(img)              # Show image on subplot
    axes[i].axis('off')              # Hide axis ticks
    axes[i].set_title(img_name, fontsize=12)  # Set image title

# Remove any unused subplots
for i in range(num_images, len(axes)):
    fig.delaxes(axes[i])

# Adjust spacing between subplots
plt.tight_layout()
plt.show()

# Run object detection on a test video
# - source="FTL test video.mp4" is the input video file
# - save=True saves the video with detections
!yolo task=detect mode=predict model=./runs/detect/train/weights/best.pt source="FTL test video.mp4" save=True
```

Results

Accurate Foreign Object Detection

- Utilized advanced computer vision models: CNNs, YOLOv8, and UNet.
- Trained models on a diverse dataset of real and synthetic runway/road images.
- Synthetic data was generated using Blender for improved model robustness.
- Accurately detected various FOD types – metal, tools, animals, debris.
- Achieved high precision in detection.

Real-Time Onboard Processing

- Trained models were deployed on the drone's onboard computer.
- Enabled real-time image processing and FOD detection during flight.
- Eliminated the need for post-flight data analysis.
- Reduced response time for runway inspections significantly.
- Enhanced overall efficiency and responsiveness of the system.

Efficient Autonomous Flight Operations

- Drone was used with manual flight paths for full runway coverage.
- Ensured complete and consistent inspection of designated areas.
- Underwent testing in both simulated and real-world environments.
- Demonstrated system stability and consistent performance.
- Proven adaptability to varied altitudes and lighting conditions.

Results

Enhanced Operational Efficiency

- Minimized reliance on manual inspections, which are labor-intensive and slow.
- Can use, 24/7 operation with consistent performance.
- Delivered faster and more efficient runway inspections.
- Increased runway availability by reducing inspection downtime.
- Offered long-term cost savings through reduced manpower and improved efficiency.

Actionable Data Collection

- System not only detected FOD but also captured detailed metadata.
- Metadata included object type, GPS location, and frequency of occurrence.
- Can provide actionable insights for airport maintenance teams.
- Supported preventive maintenance and informed operational planning.

Structured and Collaborative Development

- Camera calibration and simulation design were handled with precision to ensure optimal data input.
- Research and planning guided the system's relevance and alignment with real-world needs.
- Model training and performance tuning enhanced the accuracy of detection algorithms.
- Drone integration and testing validated the concept through extensive flight trials.

Discussion – Insights from Implementation and Results

Challenges Faced

Dataset Limitations:
Initial use of publicly available or fully synthetic datasets led to poor generalization due to unrealistic textures and object variability.
Real-world data collection was time-consuming but necessary for boosting model accuracy. This may also cause a limitation since every runway is different
Resolution of Video:
The resolution of the Video we got from in-house testing was very low to distinguish between different types of foreign objects
False Positives and Missed Detections:
Since the objective was to detect debris on the runway The model that we were using was having problems detecting foreign objects of varying size, thickness, and colour.

Key Learnings

- Building a realistic, well-labeled dataset is as crucial as model selection.
- YOLOv8's performance is strong out-of-the-box, but further domain-specific tuning can unlock even better results.
- Combining object detection with visualization code allowed for quick validation and debugging of predictions, which is essential in real-world safety-critical systems.

What Can Be Improved

- **Increase Dataset Diversity:** Collect images from different airports, lighting conditions (e.g., fog, rain, night), and camera angles. Include edge cases like partially buried FOD or objects obscured by shadows.
- **Class-wise Detection**: Instead of a single "FOD" class, label and train for sub-classes (e.g., stone, bolt, paper, plastic). This can help improve precision and enable risk categorization.
- **Increase in Real-Time FOD Efficiency:** The model Api can be generated to connect it to a server to get better time efficiency in Real-time Foreign Object Detection
- **Hyperparameter Tuning:** Explore optimization of YOLOv8's learning rate, augmentation strategy, and loss function for better precision-recall balance.
- **Edge Deployment Optimization:** Model pruning or quantization for low-power embedded devices like NVIDIA Jetson to enable real-time inference on autonomous drones.
- **Evaluation Metrics**: Implement metrics like mAP@0.5:0.95, Precision, Recall, and IoU-based heatmaps to analyze performance more deeply.

CONCLUSION

SUMMARY

We developed an AI-based runway inspection system that detects FOD using a custom-trained YOLOv8 model.
A real-world dataset, enhanced with synthetic augmentation, significantly improved detection performance over publicly available or fully synthetic datasets. After comparative model evaluation, YOLOv8 was selected for its real-time performance, robustness, and ease of deployment. Our approach addresses critical pain points in manual runway inspection – speed, accuracy, and 24/7 monitoring capability.

FUTURE WORK

- **Edge Deployment on Drones:** Optimizing the model for on-board inference on embedded hardware like NVIDIA Jetson or Raspberry Pi for full drone autonomy.
- **Multiclass Object Detection:** Expanding the system to classify types of FOD (e.g., bird, tool, plastic) for more granular risk assessment.
- **Weather Adaptivity:** Training the model under adverse weather conditions (fog, rain) to ensure consistent performance year-round.
- **Integration with Airport Infrastructure:** Creating real-time alert systems that notify ground staff and log detections into airport maintenance dashboards.
- **Long-Term Analytics:** Using FOD data over time to predict high-risk zones and suggest design optimizations in airport layout and maintenance scheduling.

Individual Learnings

Apoorva (22AE30004)

Hard skills

Blender for Synthetic Data Generation, Computer Vision and AI Model Testing, Dataset Research & Curation

Soft skills

Team Management & Coordination, Problem-Solving under Constraints, Communication & Collaborations

Vishal (22AE10046)

Hard skills

Deep learning and computer vision, YOLOv8, Synthetic data generation

Soft skills

Team collaboration, Problem solving and Resourcefulness, Project management

Mohit (22AE10023)

Hard skills

Deep learning and computer vision, YOLOv8, Labelling, Dataset Research & Curation, Real-world Problem Understanding

Soft skills

Project Management, Problem Solving, Learning Agility

Devanshi (22AE30007)

Hard skills

Annotation and Auto-labeling, Model optimisation, Blender

Soft skills

Team work, Learning agility, Project management

Vansh (223EP11)

Hard skills

Python, Machine Learning, Deep Learning, Computer Vision

Soft skills

Teamwork, Critical Thinking, Decision making, Problem solving

Harsh (22AE10012)

Hard skills

Learned drone piloting via PicaSim and camera operation based on runway dimensions,Dataset Research & Curation

Soft skills

Team work & coordination
Constraint-based problem solving
Product strategy management

Rahul (22AE10032)

Hard skills

Bird's eye camera view, Footage Post production Stabilization and color correction for processing.

Soft skills

Leadership
Problem Solving
Teamwork


21
03
21
21
03
21
THANK YOU?