野生動物保育智能追蹤系統

(Wildlife Conservation Intelligent Tracking System)

專題成員： 111B11890 陳廷翊

112B16157 李宜蓉

111B11872 蔡青錡

112B16077 余翊廷

指導老師： 陳春志

Table of Contents

1. Problem Description

1.1 Current Challenges in Wildlife Conservation

Modern wildlife conservation faces multiple complex problems:

Core Problems

- Difficulty in accurately tracking wildlife numbers and distribution
- Traditional survey methods are time-consuming and costly
- Human interference may affect animal natural behavior
- Lack of real-time, dynamic population monitoring systems
- Rapid habitat reduction

Specific Challenges

a. Complex terrain: Mountainous areas, jungles hinder traditional surveys
b. Unpredictable animal behavior
c. Limited survey personnel and budget
d. Lack of precise population data

1.2 Limitations of Existing Methods

- Manual ground surveys:
    a. Limited range
    b. High risk
    c. Non-real-time data
    d. Expensive
    e. Low efficiency
- Fixed cameras: Limited coverage area
- Satellite imagery: Insufficient resolution
- Manual counting: Prone to errors
- Traditional remote sensing technologies:
    a. Insufficient resolution
    b. Unable to precisely identify species
    c. Difficult to track animal behavior
    d. Slow data updates

2. Solution Proposal

2.1 Intelligent Drone Ecological Tracking System

System Architecture

1. Drone Hardware Platform
- High-resolution imaging equipment
- Long-lasting battery
- Precise GPS positioning
- Wind and weather-resistant design

2. AI Intelligent Analysis Module
- Real-time species identification
- Population statistics
- Geospatial analysis
- Dynamic tracking

3. Data Management Platform
- Cloud storage
- Real-time report generation
- Historical data comparison

Wildlife AI Tracker integrates:
- Drone remote sensing technology
- Generative Artificial Intelligence
- Computer Vision
- Geospatial Analysis

2.2 Key Innovations

a. AI intelligent species detection
b. Automatic path planning
c. Real-time data analysis
d. Low-interference survey mode

2.3 Core System Functions

- Autonomous drone flight

- Real-time species identification
- Population data collection
- Geographic distribution analysis
- Intelligent report generation

2.4 Expected Benefits

- Reduce survey costs by 50%
- Improve data accuracy by 80%
- Expand survey range by 300%
- Real-time ecological monitoring

3. Generative AI Solution

3.1 AI Technology Applications

Species Detection and Identification

- Using YOLO V8 Deep Learning Model

  Characteristics:

  1. Real-time detection
  2. High accuracy
  3. Multi-species identification
- Transfer learning
- Probabilistic detection algorithms

Generative AI Core Functions

1. Species Intelligent Identification

```python
def species_identification(image):
    """
    使用生成式AI進行物種辨識

    Args:
        image (np.array): 無人機影像

    Returns:
        dict: 偵測結果
    """
    # 載入預訓練模型
    model = YOLO('wildlife_model.pt')

    # 執行即時偵測
    results = model(image)[0]

    # 轉換偵測結果
    detections = {
        'species': results.names,
        'confidence': results.probs,
        'bounding_boxes': results.boxes
    }

    return detections
```

2. Dynamic Population Prediction

```python
def population_prediction(historical_data):
    """
    基於歷史數據預測未來族群變化

    Args:
        historical_data (DataFrame): 歷史族群數據

    Returns:
        dict: 族群預測結果
    """
    # 使用生成式AI進行時間序列預測
    prediction_model = TimeSeriesPredictor()

    predictions = prediction_model.forecast(
        historical_data,
        periods=12,  # 預測未來12個月
        confidence_interval=0.95
    )

    return {
        'predicted_population': predictions.mean,
        'confidence_range': predictions.confidence_interval
    }
```

3. Intelligent Ecological Habitat Analysis

```python
def habitat_analysis(spatial_data):
    """
    分析動物棲息地特徵

    Args:
        spatial_data (GeoDataFrame): 地理空間數據

    Returns:
        dict: 棲息地特徵
    """
    # 生成式AI分析地理環境
    habitat_classifier = HabitatAI()

    habitat_features = habitat_classifier.analyze(
        spatial_data,
        feature_extraction=True
    )

    return {
        'terrain_type': habitat_features.terrain,
        'vegetation_density': habitat_features.vegetation,
        'water_proximity': habitat_features.water_resources
    }
```

3.2 Intelligent Detection Process

1. Image preprocessing
2. Feature extraction
3. Species identification
4. Coordinate positioning
5. Data correlation

3.3 System Advantages

1. High-precision detection
2. Real-time dynamic tracking
3. Low-cost operation
4. Minimized ecological disruption
5. Strong scalability

3.4 Technical Innovations

- Deep learning models
- Real-time inference
- Multi-modal analysis
- Adaptive learning

3.5 Ethical and Sustainability Considerations

- Respect ecological balance
- Protect wildlife
- Support scientific research
- Minimize human interference

4. Code

```
import cv2
import numpy as np
import torch
import torchvision
import supervision as sv
from ultralytics import YOLO
import rasterio
import geopandas as gpd
from datetime import datetime
import logging

class WildlifeTrackingSystem:
    def __init__(self,
                 model_path='yolov8_wildlife.pt',
                 conservation_species=['tiger', 'elephant', 'leopard']):
        """
        初始化野生動物追蹤系統

        Args:
            model_path (str): AI 模型路徑
            conservation_species (list): 需要追蹤的保育類動物清單
        """
        # 日誌配置
        logging.basicConfig(level=logging.INFO)
        self.logger = logging.getLogger(__name__)

        # 載入預訓練物體偵測模型
        self.model = YOLO(model_path)
        self.conservation_species = conservation_species

        # 地理空間數據存儲
        self.spatial_database = gpd.GeoDataFrame()

    def process_drone_video(self, video_path):
        """
        處理無人機影像串流
```

```
        Args:
            video_path (str): 無人機影像路徑
        """
        cap = cv2.VideoCapture(video_path)

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                break

            # 執行物體偵測
            results = self.model(frame)[0]
            detections = sv.Detections.from_yolov8(results)

            # 過濾保育類動物
            wildlife_detections = self._filter_conservation_species(detections)

            # 記錄偵測結果
            self._log_wildlife_data(wildlife_detections, frame)

            # 視覺化結果
            annotated_frame = self._visualize_detections(frame,
wildlife_detections)

            cv2.imshow('Wildlife Tracking', annotated_frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

        cap.release()
        cv2.destroyAllWindows()

    def _filter_conservation_species(self, detections):
        """
        過濾特定保育類動物

        Args:
            detections (Detections): 偵測結果
```

```
        Returns:
            Detections: 篩選後的保育類動物偵測結果
        """
        filtered_detections = detections[
            np.isin(detections.class_name, self.conservation_species)
        ]
        return filtered_detections

    def _log_wildlife_data(self, detections, frame):
        """
        記錄野生動物地理空間數據

        Args:
            detections (Detections): 偵測結果
            frame (np.ndarray): 影像幀
        """
        current_time = datetime.now()

        for detection in detections:
            # 從影像中擷取地理座標(假設已整合 GPS 定位)
            latitude, longitude = self._extract_geo_coordinates(frame, detection)

            new_record = gpd.GeoDataFrame({
                'species': [detection.class_name],
                'count': [1],
                'timestamp': [current_time],
                'geometry': [Point(longitude, latitude)]
            })

            self.spatial_database = pd.concat([
                self.spatial_database,
                new_record
            ])

    def _extract_geo_coordinates(self, frame, detection):
        """
        從影像中提取地理座標(模擬方法)
```

```
        Returns:
            tuple: (緯度, 經度)
        """
        # 實際實現需要整合無人機的 GPS 模組
        return (25.0340, 121.5654)  # 預設座標

    def _visualize_detections(self, frame, detections):
        """
        視覺化偵測結果

        Args:
            frame (np.ndarray): 原始影像
            detections (Detections): 偵測結果

        Returns:
            np.ndarray: 標註後影像
        """
        box_annotator = sv.BoxAnnotator()
        annotated_frame = box_annotator.annotate(
            scene=frame,
            detections=detections
        )
        return annotated_frame

    def generate_population_report(self):
        """
        生成族群報告

        Returns:
            dict: 族群統計報告
        """
        population_report = (
            self.spatial_database
            .groupby('species')['count']
            .agg(['count', 'sum'])
            .reset_index()
        )
```

```
            return population_report.to_dict(orient='records')

# 主程式執行
def main():
    tracking_system = WildlifeTrackingSystem()
    tracking_system.process_drone_video('wildlife_footage.mp4')

    # 生成報告
    report = tracking_system.generate_population_report()
    print("族群報告:", report)

if __name__ == "__main__":
    main()
```

5. Simulated Flight Screens

https://youtu.be/qpr5SipzeMM