



CATIE



RÉGION
**Nouvelle-
Aquitaine**



UNION EUROPEENNE
Fonds Européen de
développement Régional

Outils d'imagerie pour la robotique - Sujet Projet

TS341_2025 — Robotique & Apprentissage - TS341

Version — 2025-10-01

CATIE

Sébastien Delpéuch s.delpéuch@catie.fr
Clément Pinet c.pinet@catie.fr

CATIE

Table des matières

1	Projet final TS341 - Détection de drones	3
1.1	Contexte	3
1.2	Besoin exprimé par le client	3
1.2.1	Exigences de robustesse	3
1.2.2	Démarche d'ingénieur	3
1.3	Rédaction d'un cahier des charges	3
1.3.1	Modalités de rendu	3
1.3.2	Contenu attendu	4
1.4	Réalisation d'un prototype	4
1.4.1	Structure du projet obligatoire	4
1.4.2	Responsabilités	4
1.4.3	Outils et bonnes pratiques obligatoires	4
1.5	Critères d'évaluation	5
1.5.1	Proposition technique (5 points)	5
1.5.2	Prototype fonctionnel (5 points)	5
1.5.3	Qualité du code et outils (5 points)	5

1 Projet final TS341 - Détection de drones

1.1 Contexte

Les drones civils sont de plus en plus présents dans notre espace aérien. Si leur utilisation légitime ne cesse de croître, ils représentent également des risques potentiels pour la sécurité des infrastructures sensibles, des événements publics et de la vie privée. La détection et l'identification de drones non autorisés constituent un enjeu majeur pour la sécurité.

Dans ce cadre, une entreprise spécialisée dans la sécurité des sites sensibles a fait appel au CATIE pour développer un système de détection de drones à partir d'images capturées par différents types de caméras.

Le but de ce projet pédagogique dans le cadre du module TS341 est de reproduire le travail réalisé par le CATIE.

1.2 Besoin exprimé par le client

L'objectif de ce projet est de réaliser un système de détection de drones à partir d'images et vidéos capturées par différents types de caméras. Un unique modèle de drone devra être détecté, mais dans divers scénarios :

- À différentes altitudes
- À différentes distances de la caméra
- Dans différentes conditions de luminosité
- Avec différentes technologies de caméras et objectifs

Les données d'entraînement sont disponibles via le lien suivant : [Dataset de vidéos de drones](#).

Important : toutes les données ne sont pas fournies. Certaines données sont conservées pour l'évaluation finale de votre système.

Vous devez développer un algorithme en Python permettant de détecter le drone sur des vidéos fournies en entrée et de fournir les coordonnées du drone détecté. Vous êtes libres d'utiliser n'importe quelle méthode ou technologie, qu'elle ait été vue en cours ou non.

1.2.1 Exigences de robustesse

- Le système doit fonctionner avec toutes les technologies de caméras utilisées dans le dataset
- Des faux négatifs (i.e. des drones non détectés) sont à éviter absolument
- La capacité à expliquer les échecs éventuels est tout aussi importante que la réussite

1.2.2 Démarche d'ingénieur

- Des informations techniques sont volontairement omises dans ce sujet
- Il est de votre responsabilité d'identifier les informations manquantes et de poser les bonnes questions
- La méthodologie de travail et la rigueur d'ingénieur sont au cœur de l'évaluation
- Si la détection n'est pas possible dans certains cas, vous devez être capable d'expliquer pourquoi avec des arguments techniques solides

1.3 Rédaction d'un cahier des charges

Date de rendu : 12/11/2025 18h

Le cahier des charges doit être rédigé au format Markdown dans un fichier `cahier_des_charges.md` à la racine de votre dépôt GitHub.

1.3.1 Modalités de rendu

- Envoi par mail à s.delpeuch@catie.fr avec :
 - Le lien du dépôt GitHub partagé avec [sedelpeuch](#) et [clementpinet](#)
 - Le fichier `cahier_des_charges.md` en pièce jointe

1.3.2 Contenu attendu

- Description de la compréhension du besoin
- Identification des verrous technologiques
- Description de la solution proposée long terme (si vous aviez plus de temps)
- Description du prototype qui sera réalisé (dans les heures de projet)

1.4 Réalisation d'un prototype

Date de rendu : 05/12/2025 23h59

Le dépôt GitHub sera automatiquement cloné à la date et heure du rendu. Il doit être accessible et partagé avec sedelpeuch et clementpinet.

1.4.1 Structure du projet obligatoire

Vous devez obligatoirement réutiliser la structure du projet développé pendant les cours. Votre dépôt doit contenir :

1. Un package Python utilisant Poetry comme gestionnaire de dépendances
 - Fichier `pyproject.toml` configuré avec Poetry
 - Structure de package Python standard
2. Un Dockerfile fonctionnel
 - Le point d'entrée du conteneur doit être le script principal de votre projet
 - Le conteneur doit pouvoir s'exécuter de manière autonome
 - Toutes les dépendances doivent être installées dans l'image
3. Un README.md complet
 - Instructions d'installation et d'utilisation
 - Description du script principal et de ses arguments
 - Gestion des fichiers volumineux (modèles, données) avec instructions claires
 - Documentation sur les sorties produites par votre système
4. Un document d'analyse `rapport_technique.md`
 - Description détaillée de la solution mise en œuvre
 - Méthodologie adoptée (approches testées, choix effectués)
 - Métriques d'évaluation utilisées et résultats obtenus
 - Analyse des performances (succès et échecs)
 - Discussion sur les limites et perspectives d'amélioration
 - Justification technique des choix d'architecture et d'algorithmes

1.4.2 Responsabilités

- **Définir le script principal** : c'est à vous de déterminer ce que votre script principal doit faire et comment il doit être appelé
- **Afficher les informations pertinentes** : votre système doit produire des sorties claires (détections, statistiques, visualisations, etc.)
- **Gérer les fichiers volumineux**

1.4.3 Outils et bonnes pratiques obligatoires

Les outils suivants sont obligatoires :

- **Poetry** : gestion des dépendances et packaging
- **Docker** : conteneurisation de l'application
- **Logging** : système de logs structuré (module `logging` de Python)
- **Typing** : annotations de types dans tout le code
- **PEP 8** : respect strict des conventions de style Python

Outils recommandés (bonus pour la qualité du code) :

- `pytest` : tests unitaires et tests d'intégration
- PEP 257 : documentation des fonctions (docstrings)
- CI/CD : GitHub Actions ou autre pipeline d'intégration continue
- Linters : `black`, `flake8`, `mypy`, `pylint`
- Pre-commit hooks : vérifications automatiques avant commit

1.5 Critères d'évaluation

Le projet est noté sur 15 points répartis comme suit :

1.5.1 Proposition technique (5 points)

- **Cahier des charges** : qualité de l'analyse, pertinence des choix, identification des verrous
- **Échanges informels** : capacité à poser les bonnes questions, à demander des précisions
- **Pertinence de la solution** : adéquation entre le besoin et la solution proposée

1.5.2 Prototype fonctionnel (5 points)

- **Fonctionnalité** : le système détecte-t-il effectivement les drones ?
- **Robustesse** : performance sur différentes caméras et conditions
- **Capacité d'analyse** : explication claire et technique des succès et des échecs
- **Démarche scientifique** : méthodologie d'évaluation, métriques utilisées

1.5.3 Qualité du code et outils (5 points)

- **Outils obligatoires** : Poetry, Docker, Logging, Typing, PEP 8 (pénalités si absents)
- **Outils avancés** : `pytest`, PEP 257, CI/CD, linters (bonus)
- **Architecture du code** : modularité, lisibilité, maintenabilité
- **Documentation** : README complet, docstrings, commentaires pertinents
- **Bonnes pratiques** : gestion d'erreurs, configuration, reproductibilité