Projet Long de Programmation

Une cabane à oiseaux d’observation

Meziane Reda, Hafid Yahya

Introduction

Objectif initial du projet

Concevoir une cabane à oiseaux qui, à l'aide d'une caméra et d'un RaspberryPi, détecte la présence d'un oiseau et peut le classer en fonction de ses différentes caractéristiques physiques.

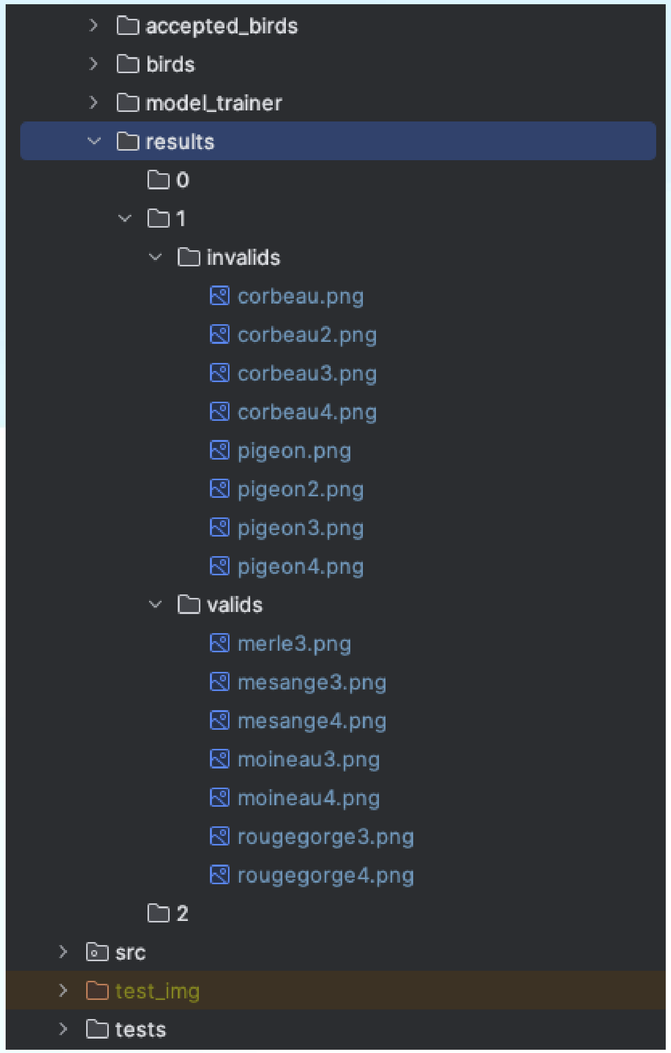
Introduction

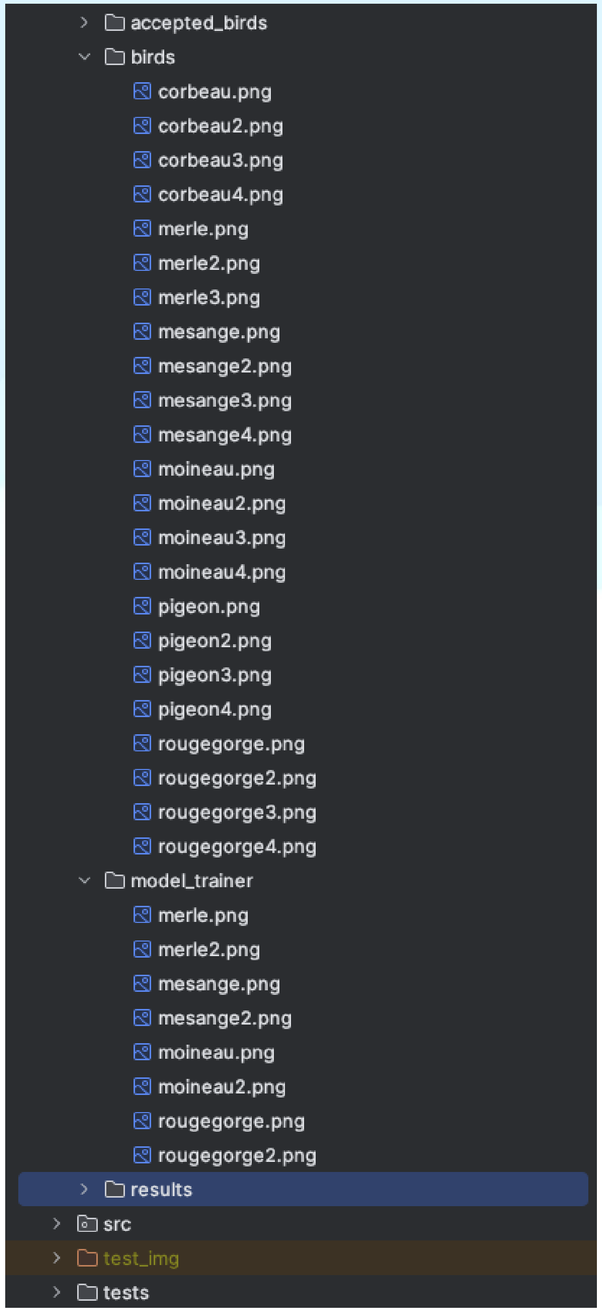
Problématique

Comment résoudre le défi de la coexistence des espèces nuisibles et non-nuisibles dans les jardins où des cabanes à oiseaux sont installées ?

Introduction

Que fait notre logiciel?





Introduction

Scénarios d’utilisation convaincants

- Vacances et on ne peut pas emmener son oiseau avec soi

- Collectionner des photos d'oiseaux

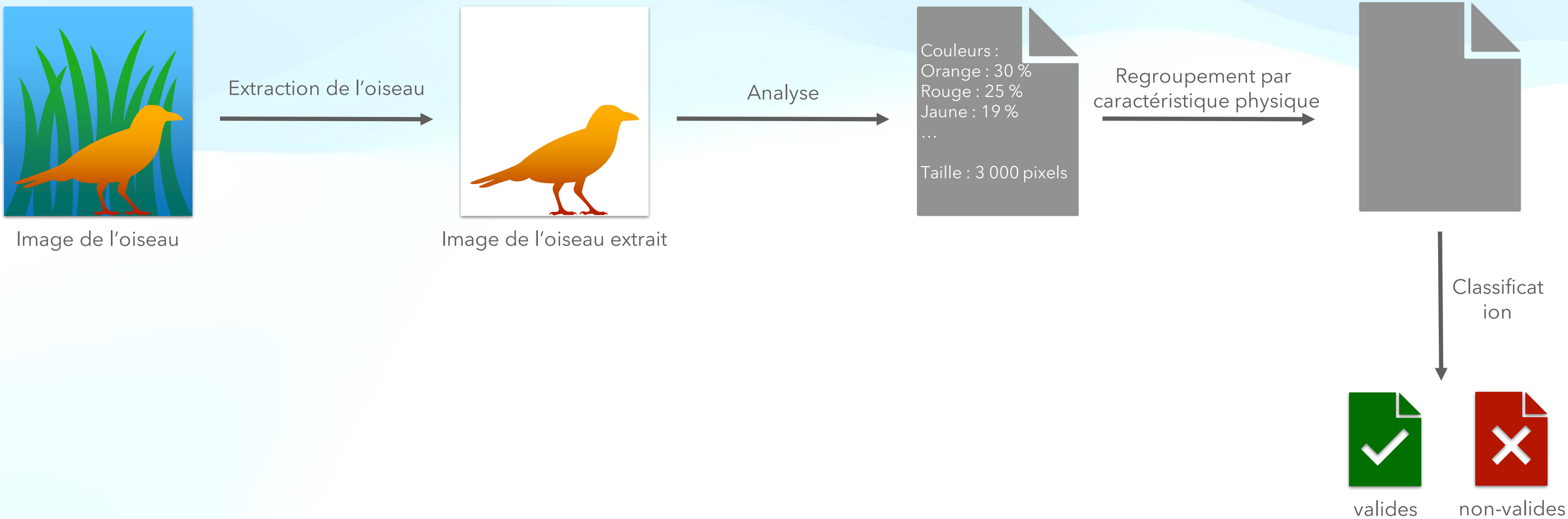
Introduction

Scénarios d’utilisation limitée

- Zones avec mélange d'espèces nuisibles
- Régions dominées par espèces nuisibles

Architecture, conception et gestion du projet

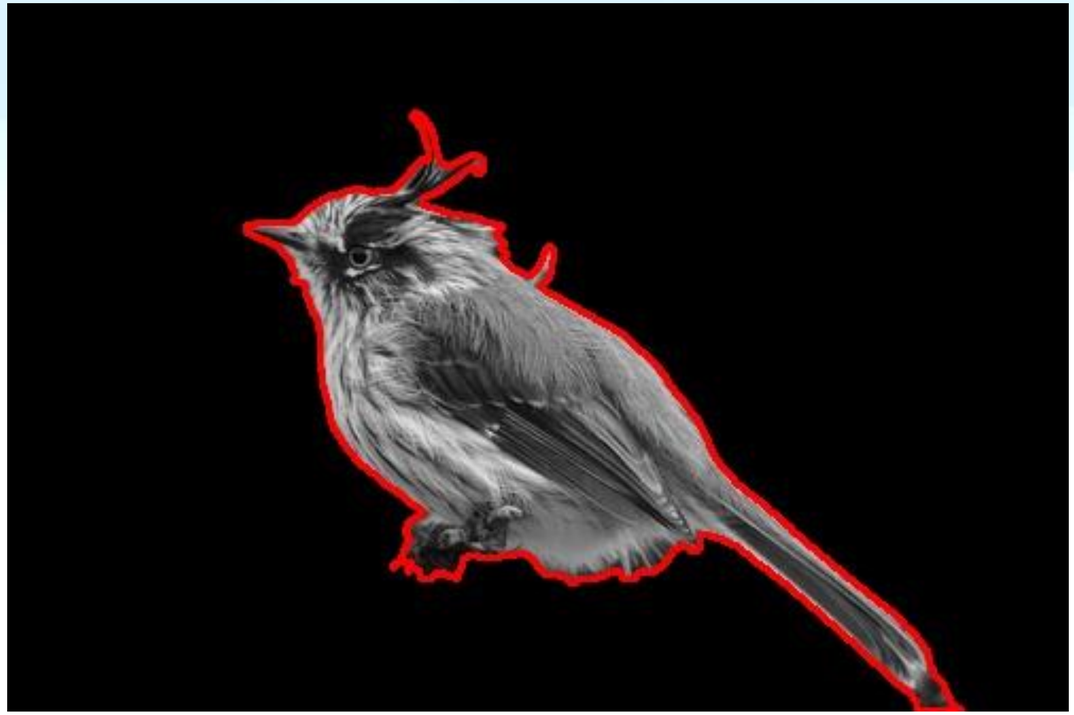
Vue d’ensemble



Architecture, conception et gestion du projet

Vue d’ensemble

Extraction de l’oiseau



Architecture, conception et gestion du projet

Choix de conception et développement

- Extraction de l'oiseau en utilisant une bibliothèque externe
- Regroupement non supervisé
- Regroupement supervisé

Étapes clés

- Algorithme de machine learning pour avoir une référence
- Extraction via notre propre algorithme
- Regroupement par couleur uniquement
- Regroupement par aire uniquement

Architecture, conception et gestion du projet

Modules principaux

- bird_extrator.py
- dimensions.py
- modele.py

Compétences techniques

- Programmation
- Traitement d'images (jeux de tests)
- Apprentissage automatique

Architecture, conception et gestion du projet

Repartition du travail

- Création des jeux de tests
- Extraction d'oiseaux
- Calcul d'aires et d'histogrammes de couleurs
- Installation Raspberry Pi
- Supervised Training
- Unsupervised Training

Architecture, conception et gestion du projet

Principales difficultés

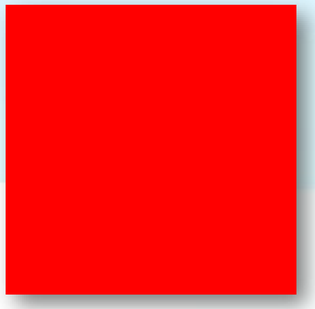
- Extraction d'oiseau
- Calcul de l'aire
- Détection de formes
- Trop abstrait, utilisation d'images fabriquées

Programmation

Algorithme Colors Count qui compte le pourcentage de présence pour chaque couleur

input :

result:



color_count_dict

colors : {120: 100.0}

color_count_dict

colors : {0: 4.43, 60: 4.28, 90: 5.6, 75: 1.76, 70: 1.12, 68: 0.98, 72: 0.72, 69: 0.71, 67: 0.38, 66: 0.66, 64: 0.43, 65: 0.54, 63: 0.36, 73: 0.69, 71: 0.65, 62: 0.26, 74: 0.64, 58: 0.1, 38: 0.09, 13: 2.19, 10: 1.63, 9: 1.21, 3: 0.52, 2: 0.41, 17: 1.02, 27: 0.25, 40: 0.1, 50: 0.1, 43: 0.09, 6: 0.56, 8: 1.03, 7: 0.65, 4: 0.52, 1: 0.31, 5: 0.56, 21: 0.79, 78: 0.44, 30: 0.38, 42: 0.09, 55: 0.1, 179: 0.25, 11: 1.92, 177: 0.5, 178: 0.48, 77: 0.43, 76: 0.48, 53: 0.14, 12: 2.04, 80: 0.43, 18: 1.12, 14: 1.69, 52: 0.08, 25: 0.35, 56: 0.13, 132: 0.14, 162: 0.12, 84: 0.29, 150: 0.34, 164: 0.12, 173: 0.48, 171: 0.3, 166: 0.13, 167: 0.2, 176: 0.54, 175: 0.42, 174: 0.38, 15: 1.46, 170: 0.31, 172: 0.22, 168: 0.26, 45: 0.15, 169: 0.2, 163: 0.14, 154: 0.1, 161: 0.11, 159: 0.13, 148: 0.04, 147: 0.08, 155: 0.1, 146: 0.1, 144: 0.07, 143: 0.13, 157: 0.05, 145: 0.07, 141: 0.12, 136: 0.1, 120: 0.63, 160: 0.14, 140: 0.12, 137: 0.1, 165: 0.27, 138: 0.12, 134: 0.11, 135: 0.21, 153: 0.06, 129: 0.13, 130: 0.13, 139: 0.09, 128: 0.19, 158: 0.14, 156: 0.07, 95: 0.67, 133: 0.15, 142: 0.05, 16: 1.09, 125: 0.17, 19: 1.06, 152: 0.03, 124: 0.14, 20: 1.02, 23: 0.62, 28: 0.22, 26: 0.3, 22: 0.57, 24: 0.35, 29: 0.12, 33: 0.09, 105: 0.97, 113: 0.82, 117: 0.32, 131: 0.13, 107: 1.87, 126: 0.16, 116: 0.41, 32: 0.07, 109: 1.98, 108: 2.2, 123: 0.2, 110: 1.7, 100: 0.15, 111: 1.37, 81: 0.25, 115: 0.43, 114: 0.54, 106: 1.32, 51: 0.1, 48: 0.09, 83: 0.35, 34: 0.07, 54: 0.1, 49: 0.09, 46: 0.08, 39: 0.09, 44: 0.1, 47: 0.09, 59: 0.1, 41: 0.08, 37: 0.07, 79: 0.25, 31: 0.06, 61: 0.1, 35: 0.08, 57: 0.12, 36: 0.08, 82: 0.21, 85: 0.34, 86: 0.44, 87: 0.67, 89: 2.72, 91: 4.17, 99: 0.11, 93: 1.32, 92: 2.59, 96: 0.18, 94: 1.01, 97: 0.08, 88: 2.1, 98: 0.15, 103: 0.29, 112: 0.92, 127: 0.11, 122: 0.17, 102: 0.21, 121: 0.11, 118: 0.34, 119: 0.18, 101: 0.17, 104: 0.51}

Programmation

Algorithme

Comparaison du pourcentage de similitudes des deux images (couleurs et aires).

Testabilité

Tester le bon fonctionnement de votre projet

CI/CD

Images fabriquées artificiellement.

Test en condition réelle.

Testabilité

Éléments factuels

Images classées correctement à 80%

Limites d'utilisation

Distance de l'oiseau par rapport à la caméra.

Présence de plusieurs oiseaux en même temps.

Espèces qui se ressemblent.

Luminosité ambiante.

Conclusion

- Ce que nous avons appris :

Mettre en place un algorithme de Machine Learning, Implémentation d'algorithmes sur un RaspberryPi.

- Version 2.0 :

Capteur Laser Infrarouge pour détecter la distance de l'oiseau par rapport à la caméra.

Dispositif de dissuasion des espèces nuisibles

Autres espèces

Detection de chant pour plus de précision

- Ce que l'on aurait fait différemment :

Se concentrer plus tôt sur la partie matérielle et les tests en conditions réelles.