PROJET DRONE :
Détection et suivie d’un objet en mouvement

Félix Capuano Yohan Anoux Carole Ravel Theodore Lerondeau

Sommaire :

I. Présentation du sujet

Le but de ce projet et de réaliser un algorithme basé sur les différentes fonctionnalités présentes sur le drone.

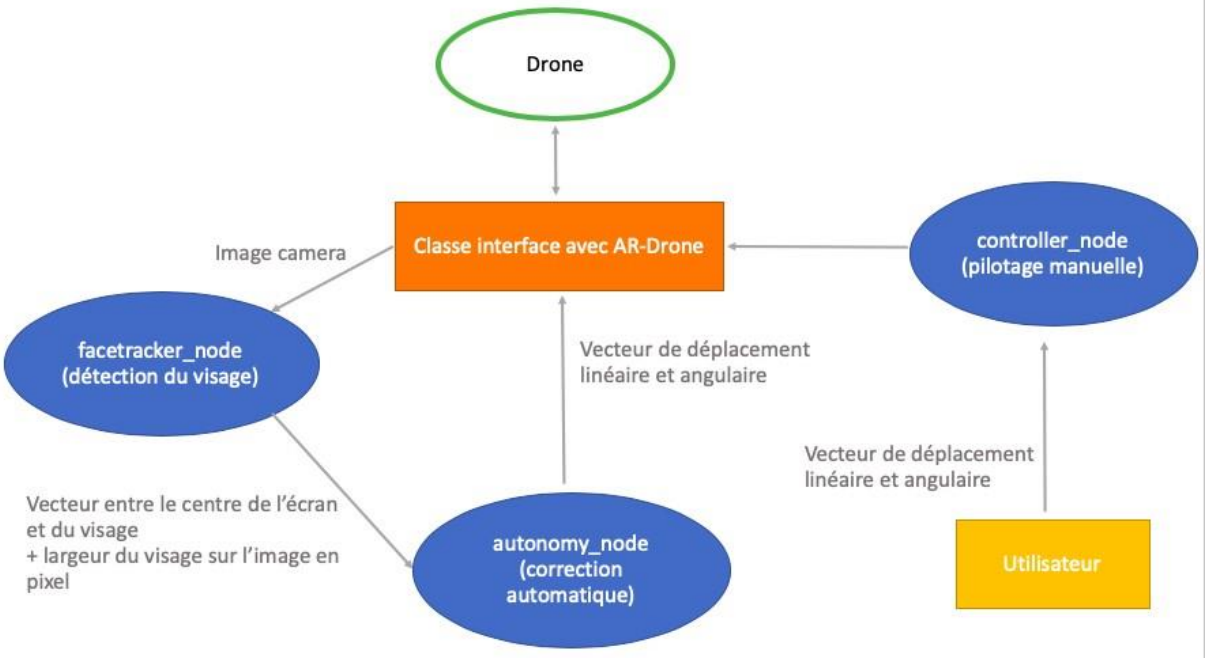
Après consultation le groupe à décider de créer un algorithme permettant de suivre un visage détecté grâce à la caméra frontale du drone. Le drone devra pouvoir suivre le visage en le gardant au centre de sa vision mais aussi en le suivant si celui-ci tourne d'un côté ou de l'autre et s'il s'éloigne ou se rapproche.

Nous avons utilisé, la plateforme de développement logicielle ROS (Robot Operating System) pour créer nos programmes nous permettant de communiquer et contrôler avec notre drone. Nous avons également pu créer un environnement virtuel pour la commande du drone grâce à Gazebo. Cet outil nous a été très utilise pour nos phases de test.

II. Architecture du projet

Notre projet est constitué premièrement, d'un classe « AR-Drone » permettant d'établir la connexion et interagir avec le drone.

Il est ensuite constitué de trois nœuds : Facetracker_node qui permet la détection d'un visage, et autonomy_node et controller_node permettant de mettre en mouvement le drone. autonomy_node permet au drone de corriger ses mouvements de manières automatiques lorsque la personne se déplace. controller_node permet à l'utilisateur de pouvoir prendre le contrôle du drone. Cela peut être utile pour repositionner le drone s'il permet sa cible. (Très utile pour les phases de test)



III. La détection de visage

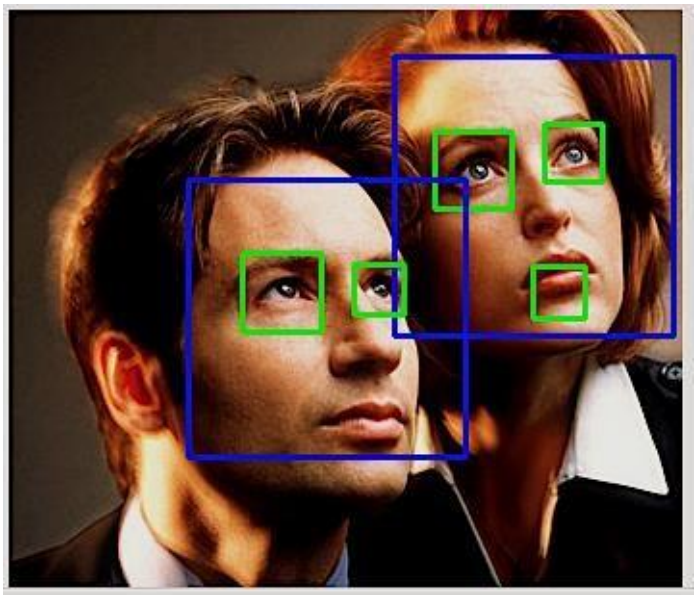
Ici nous allons confronter 2 modèles de détections de visages et les comparer pour choisir le plus performant.

1. Premier algorithme : Haarcascade Classifier

Dans cette partie du travail, nous avons confronter 2 algorithmes.
Dans un premier temps nous avons mis en place un algorithme basé sur Haarcascade Classifier :

- Les librairies OpenCV "Haarcascade" sont des algorithmes de détection d'objets en Machine Learning.
- Ils utilisent les caractéristiques de Haar pour déterminer la probabilité qu'un certain point fasse partie d'un objet.

- Ces librairies sont pré-entrainées dans des fichiers .xml et prêtes à l'emploi.



```python
import numpy as np
import cv2
import os

dir_path = os.path.dirname(os.path.abspath(__file__))

xml_data = dir_path + "/../resources/cv2/haarcascade_frontalface_default.xml"
face_cascade = cv2.CascadeClassifier(xml_data)

def get_heads_pos(frame):

    faces = face_cascade.detectMultiScale(frame, 1.3, 5)
    centers = []

    for (x,y,w,h) in faces:
        centers.append((x + w//2, y + h//2))

    return centers
```

On importe la librarie "haarcascade_frontalface_default.xml pour la detection de visage.

detectMultiScale() : Détecte les objets de différentes tailles dans l'image d'entrée. Les objets détectés sont renvoyés sous la forme d'une liste de rectangles.

Calcul du centre de l'objet detécté.

Les résultats obtenus avec cet algorithme sont concluant néanmoins, pour permettre un meilleur travail pour des futurs potentiels améliorations nous avons essayé un autre algorithme basé sur Dlib :

2. Deuxième algorithme : La librairie Dlib

Dlib est une bibliothèque logicielle multi-plateforme à usage général écrite dans le langage de programmation C++.
Nous utilisons la fonction get_frontal_face_detector() qui renvoie renverra un détecteur qui récupère les informations sur les visages.
Chaque visage est un objet qui contient les points où l'image peut être trouvée.

```python
import cv2
import os
import dlib

dir_path = os.path.dirname(os.path.abspath(__file__))
#path = dir_path + "/../resources/cv2/shape_predictor_68_face_landmarks.dat"
#predictor = dlib.shape_predictor(path)

detector = dlib.get_frontal_face_detector()

def get_heads_pos(frame):

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = detector(gray)

    center = None
    max_face_size = 0
    face_target = None
    for face in faces:
        face_left = int(face.left())
        face_right = int(face.right())
        current_face_size = abs(face_right-face_left)

        if current_face_size > max_face_size:
            max_face_size = current_face_size
            face_target = face

    x, y = 0, 0
    if face_target != None:
        x1 = int(face_target.left())
        y1 = int(face_target.top())
        x2 = int(face_target.right())
        y2 = int(face_target.bottom())

        x = int((x1 + x2)/2)
        y = int((y1 + y2)/2)

    return (x, y, max_face_size)
```

On importe la librarie "get_frontal_face_detector" pour la detection de visage.

On calcule la distance maximale des visages detectés sur la frame

On ne détecte que le visage qui est le plus proche de la caméra

On récupère les 4 coins du rectangles du visage detécté.

On récupère les coordonnées du centre du rectangle ainsi que la taille maximale entre la gauche et la droite du visage

Notre choix c'est tourné vers la detection de visage avec la librarie **DLib**.

Pourquoi ?

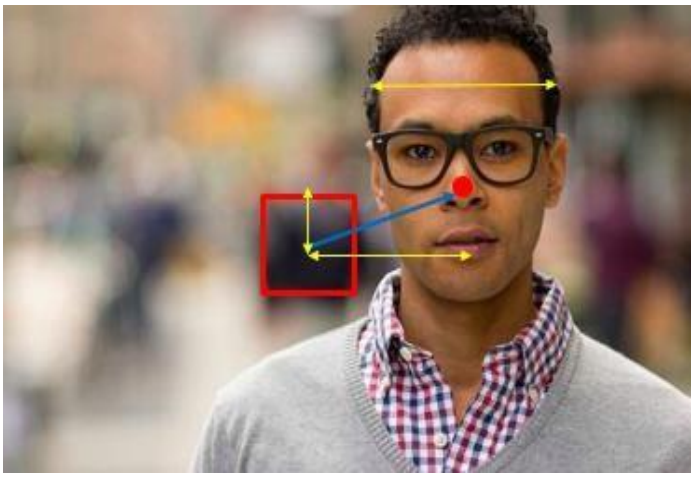
- Le script de detection avec DLib est nettement plus précis que celui de la librairie HaarCascades.

- Le script de detection avec HaarCascades permet de detecter plusieurs visages à la fois, ce qui est imcompatible avec le bon fonctionnement de notre algorithme via le drone.
- L'implémentation de la fonction get_frontal_face_detector est très simple d'utilisation et est moins coûteu se en ressource que d'importer le fichier xml "haarcascade_frontalface_default"

IV. Mise en mouvement du drone

Pour mettre en mouvement notre drone, nous avons seulement eu besoin des coordonnées du centre du visage.

Le rectangle étant la zone d'acceptabilité.
Un vecteur est tracé entre le centre du visage et le centre de l'image.



On compare les composantes du vecteur aux limites de la zone centrée.
On peut mouvoir le drone en conséquence :

- D'abord la hauteur
- Puis la position horizontale

On calcule la largeur de la tête pour mouvoir le drone en avant ou en arrière pour rester à distance.

Pour anticiper la rotation du visage d'un côté ou de l'autre, lorsque le drone se déplace vers la droite, on faire tourner le drone sur lui-même dans le sens contraire

V. Améliorations possibles

Des ameliorations sur la detection du visage et sur les mouvements du drone aurais pu être apportés.

Ameliorations sur la detection du visage :

- Possibilité de pouvoir détecter une rotation du visage
- Instaurer de la reconnaissance faciale

Améliorations du mouvement :

- Rotation plus précise autour d'une personne
- Accélération et décélération progressive

VI. Conclusion

A travers l'ensemble de ce projet nous avons pu parcourir diverses méthodes de tracking de visage ainsi que diverses méthodes pour mouvoir le drone.
Grace à l'utilisation de gazebo nous avons pu tester ces divers algorithmes dans une simulation et nous rendre compte des potentiels faiblesses que ceux-ci peuvent apporter.