

ПРОГРАММА

ПРОФЕССИОНАЛЬНОЙ ПЕРЕПОДГОТОВКИ

**“Python-разработка для БПЛА: паттерны проектирования, API-интерфейсы и библиотеки для оптимизации решений”**

**Задание 14.**

**Интеграция машинного обучения в управление БПЛА**

Преподаватель курса

**ФИО**

Выполнил

**Зубков Евгений Юрьевич**

**№ потока БПЛА\_256-1, группа 1**

**Москва 2024 г.**

**Задание 14. Интеграция машинного обучения в управление БПЛА**

**Инструкция:**

Шаг 1. Подготовить обучающую выборку данных - набор фотографий местности с разметкой (где можно совершить посадку, а где нет).

Шаг 2. Используя библиотеку Scikit-Learn, обучить модель классификации изображений (например, SVM или решающее дерево) для определения пригодных мест для посадки.

Шаг 3. Получить с камеры БПЛА изображение местности, применить обученную модель для классификации изображения и определения, можно ли совершить посадку в данном месте.

Шаг 4. Использовать результат классификации для принятия решения о посадке БПЛА.

**Модель:**

**Изображение с камеры:**

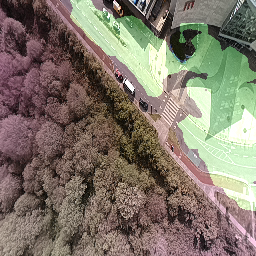


**Для обучения:**

 **можно сажать**

 **нельзя сажать**

 **нельзя сажать**

 **нельзя сажать**

 **можно сажать**

**Код:**

import numpy as np  
import cv2  
from sklearn.model\_selection import train\_test\_split  
from sklearn import svm  
from sklearn.metrics import accuracy\_score  
  
  
def load\_data(images, labels):  
 data = []  
 for img in images:  
 # Чтение изображения  
 image = cv2.imread(img)  
 # Изменение размера изображения  
 image = cv2.resize(image, (64, 64))  
 # Преобразование в одномерный массив  
 data.append(image.flatten())  
 return np.array(data), np.array(labels)  
  
  
# загрузка изображений и меток (True = можно сажать, False = нельзя сажать)  
images = ['img2.png', 'img3.png', 'img4.png', 'img5.png', 'img6.png']  
labels = [True, False, False, False, True]  
X, y = load\_data(images, labels)  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Обучение модели  
model = svm.SVC()  
model.fit(X\_train, y\_train)  
  
  
# Получение изображения с камеры БПЛА (в реальном времени)  
def get\_uav\_image():  
 image = cv2.imread('img.jpeg') # Здесь должно быть реальное изображение с камеры  
 image = cv2.resize(image, (64, 64))  
 return image.flatten()  
  
  
uav\_image = get\_uav\_image()  
# классификация изображения  
prediction = model.predict([uav\_image])  
  
  
def make\_landing\_decision(prediction):  
 if prediction:  
 print("Место пригодно для посадки.")  
 else:  
 print("Место непригодно для посадки.")  
  
  
make\_landing\_decision(prediction)  
  
# Оценка модели  
y\_pred = model.predict(X\_test)  
accuracy = accuracy\_score(y\_test, y\_pred)  
print(f"Точность модели: {accuracy \* 100:.2f}%")

**Вывод:**

**C:\DATA\BPLAPractise\pythonProject\pythonProject1\.venv\Scripts\python.exe C:\DATA\BPLAPractise\pythonProject\pythonProject1\homework\_14\_1.py**

**Место пригодно для посадки.**

**Точность модели: 100.00%**

**Process finished with exit code 0**