Capstone Project
Machine Learning Engineer Nanodegree

Andrejs Zujevs
April 21st, 2017
Riga

Quadcopter Automatic Landing Detection

I. Definition

Project Overview

Drones are more formally known as unmanned aerial vehicles. Essentially, a drone is a flying robot and a quadcopter is a specific type of drone, propelled by four motors (see Figure 1). Quadcopters and other multicopters can fly autonomously for different purposes, for example, to deliver a pizza. Usually an autonomous drone's take-off is from a home position (turn-on the motors and rise up into the air – in this project described as drone's status – 'Take-off'), flying to a defined destination point (described as drone's status – 'Flight'), then returning back to the home location and landing (described as drone's status – 'Land'). In this project the word 'landing' means the act of drone coming to land/surface and then turning off the motors in time (immediately with touching the land surface).

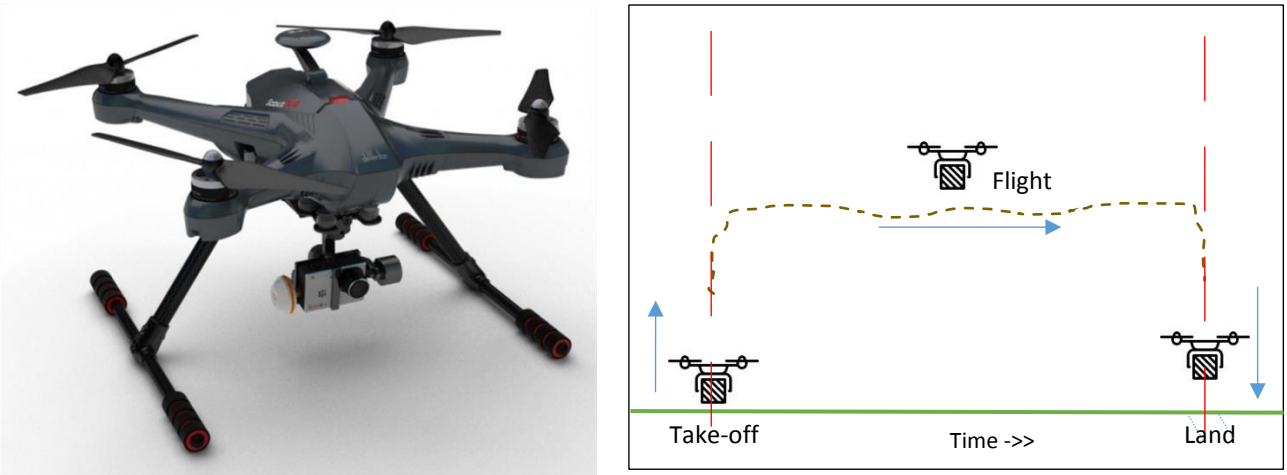

Figure 1. Quadcopter (from: http://www.hobbyflip.com) (left), drone' states (right)

In reality, the autonomous landing isn't easy for a drone due to the noise in sensors. Usually drones use one single sensor or a combination of them to measure the actual distance to the land surface and then turn-off the motors if it touches the land surface or is very close to it. For example, ultrasound, LIDAR and optical sensor (stereo vision) is mostly used for measuring the distance to land on a surface. There exist common problems of the drone autonomous landing:

1) noise in the sensor(-s) due to the impact of the other electronics in a drone;
2) different surfaces of the landing location; sensors may measure different altitude, although in reality the drone is at the same altitude;
3) slope of the landing surface and the drifting of the drone to other location;
4) inaccuracy in detecting the location of the landing caused by the inaccuracy of GPS sensor or low GPS signal;
5) dirt on the sensors (dust, water, etc.) due to environment conditions.

The above mentioned problems will cause a headache for drone developers and users:

1) drone's propellers may be damaged, due to its drifting to a side and collision with an obstacle or landing in a high grass which in turn damage the propellers;

2) drifting of a drone (propellers not turned-off in time) may cause an injury (cut, bruise, etc.) to the user, other people or animals;
3) a drone may cause damage to the property of a third party if the drone landing position is detected with inaccuracy due to GPS signal problem or noise in sensors.

The dataset available for this project consists of 25 short time flights (with one or many landing events during a flight). The total size of the dataset is ~88K records with 21 feature (different sensors' data and derived data from sensors as well as data from motors). Large size of the dataset is explained with high frequency of data recording (approximately for every 30 millisecond). Each flight has three categories of data records which belong to the drone's states: Take-off, Flight and Land (accordingly to example in Figure 1). The sequence of the drone statuses may be repeated many times if a drone landed or landing canceled more than once during a flight. Also, land surface touch moment (time interval ~0.5 sec) is manually labeled for each landing event than drone's status is Land.

Problem Statement

The aim of this project – to create an algorithm/approach which will detect the time moment of a drone touching a surface during the landing. The final algorithm should use only actual/current data from sensors and motors **without using** data from the external sensors such as LIDAR, ultrasonic and vision sensor due to the previously mentioned problems with them. Also, the final algorithm should be used in real time with limited computation resources.

The defined problem may be solved using supervised learning due to the touch moment is labeled and we only need to construct a classification model that can predict the touch moment with the land surface.
First, we need to explore the dataset, then detect outliers and remove theirs if necessary. Factor Analysis (FA) or Independent Component Analysis (ICA) will reduce number of features and then we will construct classification model by using K-Means or GMM (Gaussian Mixture Models).

Metrics

In the supervised learning case we need to take into account FP (False Positives) and FN (False Negatives), because FP will prematurely turn-off motors and the drone may crash from high altitude, then again FN will delay motors turn-off and drone may drift and/or cause injury to people or animals. Finally, more appropriate metric is F1-score due to it's a weighted average of the precision and recall.

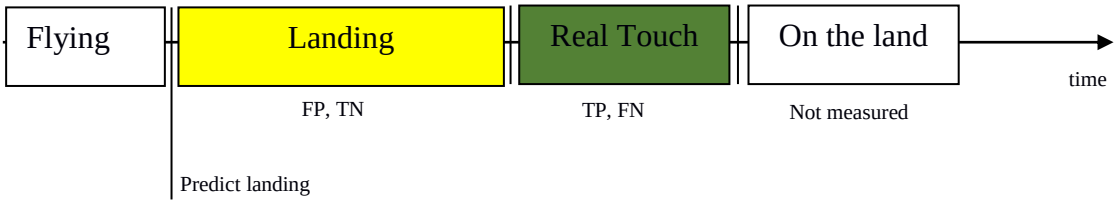

Figure 2. Predicting a touch moment during the drone landing

II. Analysis

Data Exploration

The dataset consists of 88084 records with 21 feature:

```
BAR             float64 # Barometer's sensor data
MOTOR1            int64 # Rotation speed of motor 1, rpm
MOTOR2            int64 # Rotation speed of motor 2, rpm
MOTOR3            int64 # Rotation speed of motor 3, rpm
MOTOR4            int64 # Rotation speed of motor 4, rpm
LIDAR           float64 # LIDAR's sensor data, m
THRUST          float64 # Thrust of motors (mixed), rate
ACC_X           float64 # Accelerometer's sensor data (acceleration in X axis) m/s^2
ACC_Y           float64 # Accelerometer's sensor data (acceleration in Y axis) m/s^2
ACC_Z           float64 # Accelerometer's sensor data (acceleration in Z axis) m/s^2
MAG_Z           float64 # Magnetometer's sensor data (magnetic flow in Z axis), rate
GYRO_X          float64 # Gyroscope's sensor data (orientation in X axis), rate
GYRO_Y          float64 # Gyroscope's sensor data (orientation in Y axis), rate
GYRO_Z          float64 # Gyroscope's sensor data (orientation in Z axis), rate
PITCH           float64 # Drone pitch - internally estimated using different sensors
ROLL            float64 # Drone roll - internally estimated using different sensors
YAW             float64 # Drone yaw - internally estimated using different sensors
TIME            float64 # Log's record time in microseconds from logging started
LOG_ID            int64 # Log file ID
LAND_STATUS       int64 # Status- is drone actually touched a surface or not (in ~0.5s
time frame)
STATUS           object # Drone's state
```

Samples of the dataset:

	BAR	MOTOR1	MOTOR2	MOTOR3	MOTOR4	LIDAR	THRUST	ACC_X	ACC_Y	ACC_Z	MAG_Z	GYRO_X	GYRO_Y	GYRO_Z	PITCH	ROLL	YAW	TIME	LOG_ID	LAND_STATUS	STATUS
1	1019.44	1298	1298	1298	1298	0.047	0.000	-0.906	0.055	-9.837	0.489	0.015	0.027	-0.003	-0.081	0.016	2.633	64927667	0	0	take_off
100	1019.29	1517	1624	1507	1531	0.298	0.686	-0.116	-0.083	-13.008	0.457	-0.221	0.003	-0.055	0.083	-0.015	2.614	66566750	0	0	take_off
200	1018.91	1437	1486	1445	1466	3.263	0.521	0.334	-0.291	-9.822	0.483	-0.008	0.057	-0.003	0.001	0.033	2.628	68336653	0	0	take_off
1073	1019.23	1425	1515	1396	1481	0.495	0.512	-0.498	-0.075	-10.002	0.474	0.039	0.024	-0.035	-0.056	0.021	-1.934	84077889	0	1	land
9999	1019.35	1513	1616	1509	1572	0.177	0.622	-0.233	-0.001	-12.335	0.463	-0.156	-0.133	-0.032	0.090	-0.011	3.000	42166481	5	0	take_off
32050	1012.42	1535	1525	1461	1462	3.150	0.546	0.091	0.199	-8.652	0.498	0.016	0.054	-0.022	0.002	-0.031	-1.020	550829712	10	0	flight

The dataset consists from data time series where each belongs to the particular drone flying event. The feature's TIME values are in microseconds. The records' TIME when the drone is in status TAKE-OFF is with some shift due to a log file which started writing a little earlier before the take-off. Also you can see that the speed of the motors is represented with integer numbers (rpm), LIDAR (0 m – 10 m calibrated) and BAR sensor provides their raw scalar values. Thrust is scalar and in range [0.0, 1.0], there 0.0 is minimum value and motors' rotation speed for that is 910 rpm.

Descriptive statistics of the dataset:

	BAR	MOTOR1	MOTOR2	MOTOR3	MOTOR4	LIDAR	THRUST	ACC_X	ACC_Y	ACC_Z	MAG_Z
count	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000	88084.000
mean	1014.537	1466.298	1510.502	1441.252	1476.070	5.890	0.537	-0.064	-0.064	-9.802	0.482
std	2.757	97.355	100.373	99.777	99.054	8.340	0.105	0.659	0.528	1.357	0.021
min	1009.240	910.000	910.000	910.000	910.000	0.000	0.000	-20.925	-7.646	-47.693	-0.519
25%	1012.170	1434.000	1480.000	1406.000	1445.000	0.059	0.509	-0.378	-0.361	-10.153	0.472
50%	1013.980	1477.000	1517.000	1450.000	1483.000	4.887	0.546	-0.057	-0.097	-9.775	0.484
75%	1017.410	1514.000	1559.000	1492.000	1526.000	7.956	0.585	0.235	0.191	-9.383	0.494
max	1019.680	1879.000	1879.000	1819.000	1878.000	65.324	1.000	16.850	15.148	9.816	0.561

From the descriptive statistics you can see that BAR sensor's values are in the range 1009-1019, meaning that flight evens recorded in the same place and this sensor's value is relative to place where drone flied. LIDAR sensor' values are in the range 0.0-65.32, although sensor precisely can measure altitude in range 0.2 – 10 m and the smaller/larger values are out of range of sensor's sensitivity. The feature ACC_Z values are mostly negative due to difference of drone acceleration in Z axis and gravity constant g=~9.8m/s^2.

Exploratory Visualization

In the figures below depicted sensors' values from the log ID=1. The Figure 3 represents motors' and barometer sensor values. As you can see barometer's value significantly decreased after the 80 second (drone is taking-off) and increased after the 185 second (drone is landing). In the 201 second the drone touched the land surface and all four motors after that time point were in relatively low rotation speed. For the visualization reasons the BAR sensor's values shifted by -1.002+5e and scaled by 100.0.

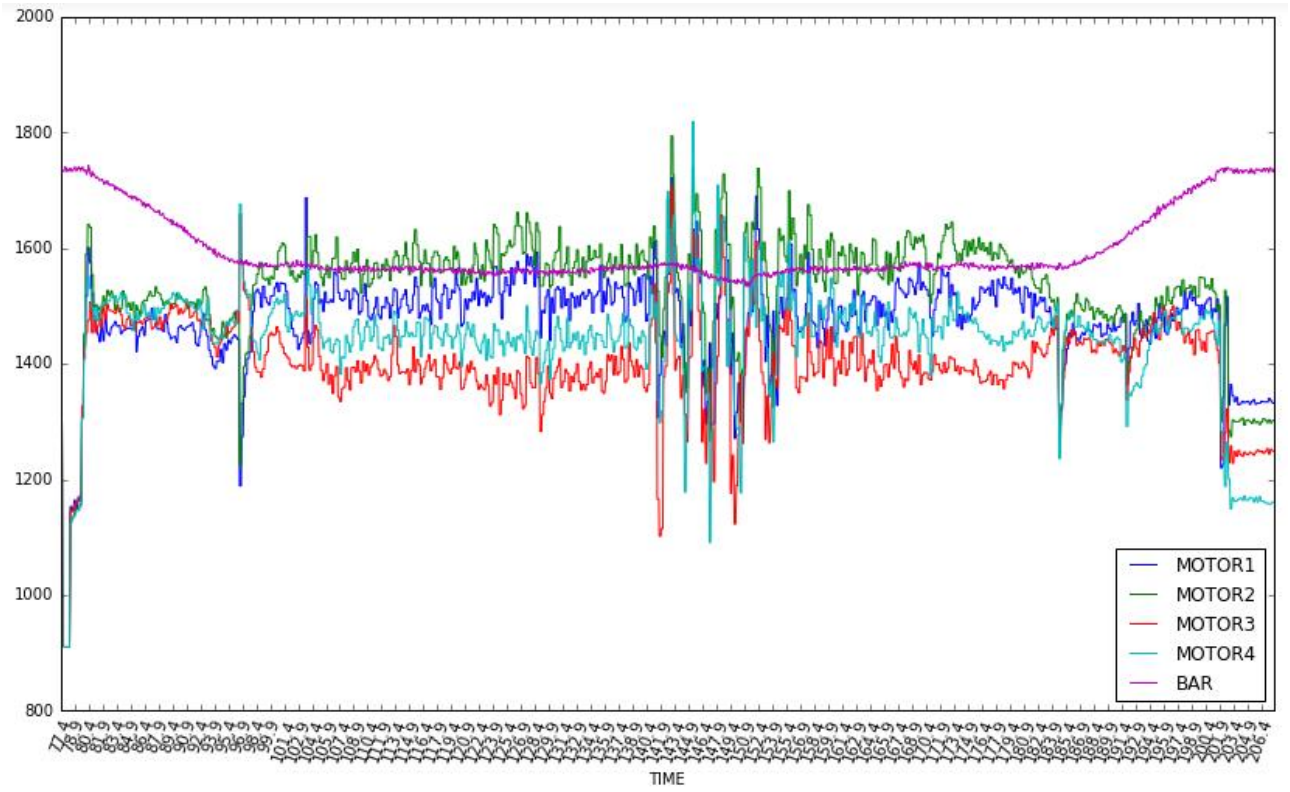

Figure 3. Log ID1's motors and barometer data during take-off, flight and landing

In the Figure 4 values of the feature ACC_Z (accelerometer' value for the z-axis) are shifted by 8.0 (for the visualization reason). The land touching time moment happened in the 201 second and where you can see high spikes for ACC_X and ACC_Y feature.

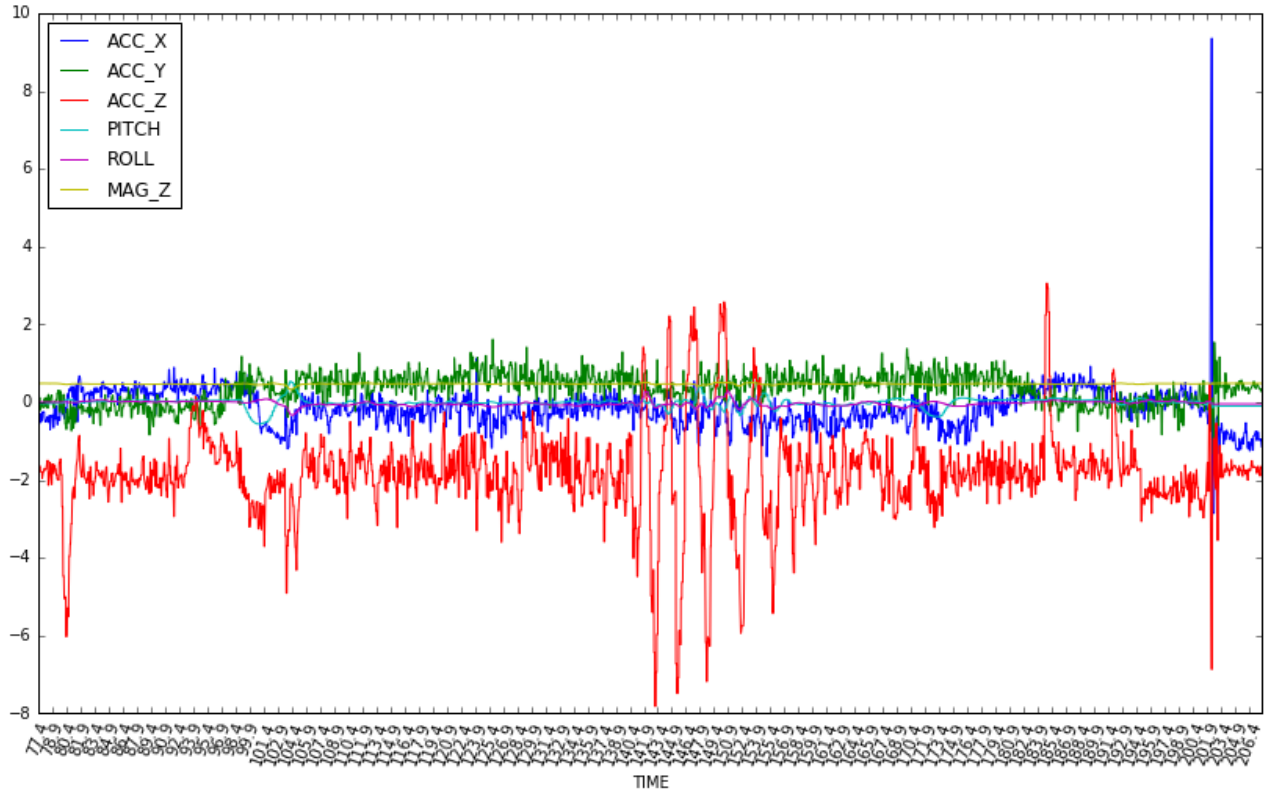

Figure 4. Log ID1's accelerometer and gyroscope data during take-off, flight and landing

In the next figure LIDAR sensor's data presented (also for log ID=1), where sensor's values scaled using 1.5 rate (for visualization reasons). Abundance of spike lines explained by sensor calibration range, where altitude above 12 m interpreted by sensor incorrectly.

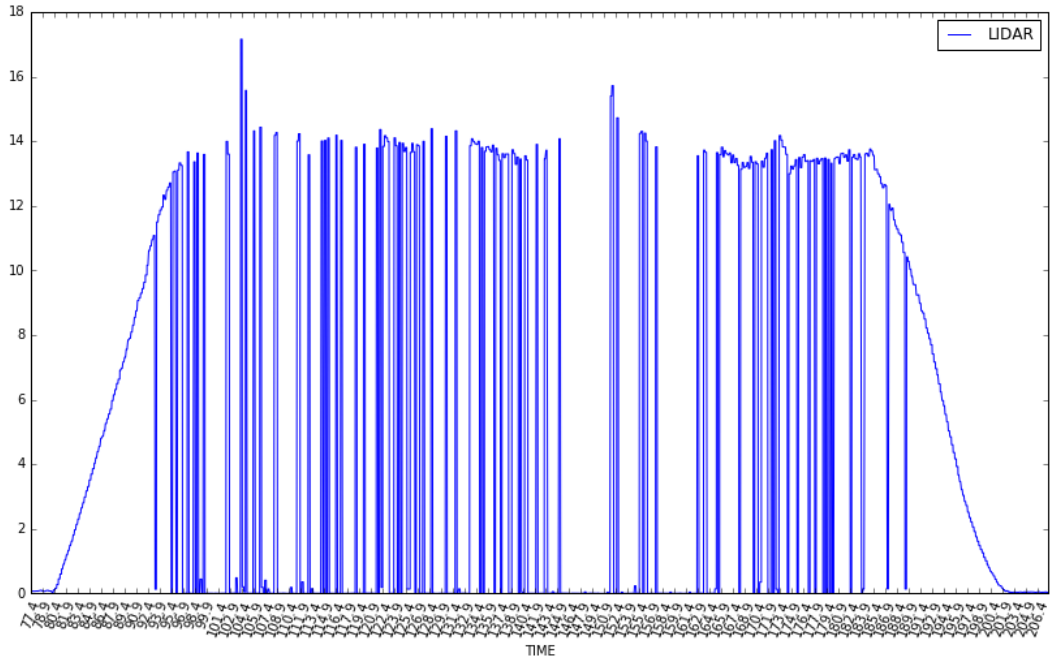

Figure 5. Log ID1's LIDAR sensor data during take-off, flight and landing

In the figures below represented scatter plots with estimation of each feature values' frequencies in the diagonal and data correlation between different features.

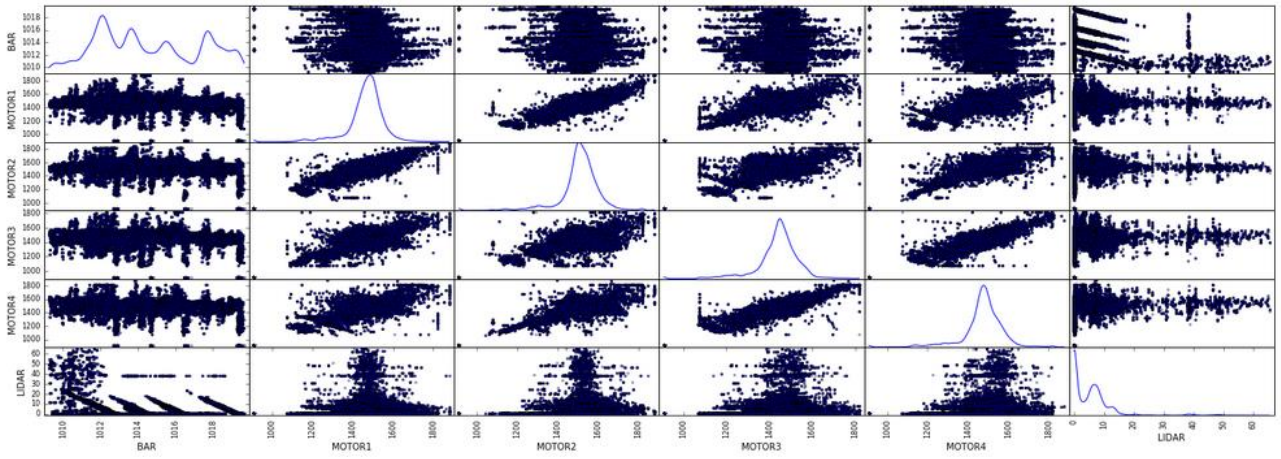

Figure 6. KDE plot for BAR, MOTOR[1-4] and LIDAR

In the figure above some correlation of motors exists due to motors rotations are synchronized by flight controller.

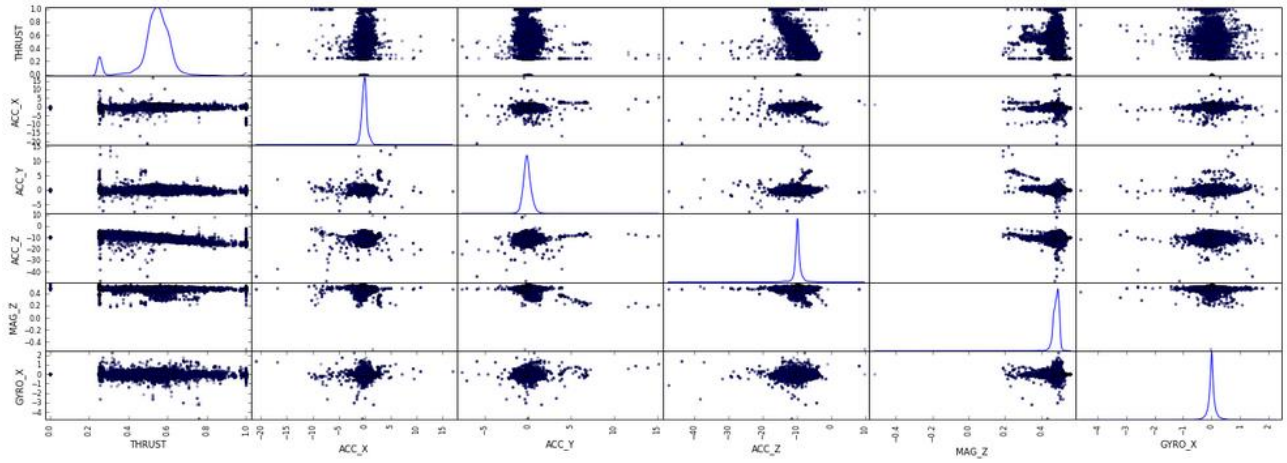

Figure 7. KDE plot for TRUST, accelerometer, magnetometer and gyroscope

In the figure above some correlation present between the THRUST and ACC_X, ACC_Y, ACC_Z feature.

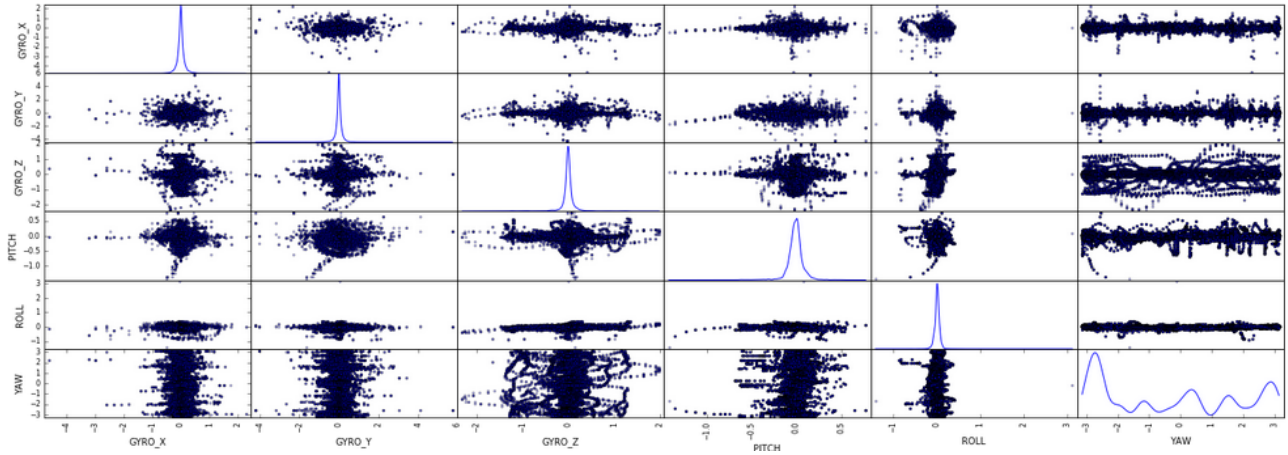

Figure 8. KDE plot for gyroscope, pitch, roll and yaw

In the figure above some correlation exists between all features due to PITCH, ROLL and YAW controlled by the drone flight controller and the gyroscope state depends on it.

Algorithms and Techniques

Firstly, data cleared from noise using outlier detection technique and data visual analysis. Then data normalized using logarithm function, due to it some features were scaled to positive values range. Secondly, using features relevant analysis, were more relevant features used for further analysis. Due to the need to find approach, how to detect the drone touch moment with the land surface then the drone is in state – landing, the ICA and Factor Analysis were used. Finally, the Factor Analysis was used prior to K-Means clustering was applied.

Dimension reducing – ICA was applied on the dataset to reduce dimensionality, but ICA requires normal distributed values and Shapiro-Wilk test approved that features' data are not normal distributed. Hence, FA used for dimensionality reducing.

Clustering – K-means and GMM clustering methods were used due to they are simple, fast and most used clustering algorithms. They are scales well on large datasets and uses distances between points. For the project problem, we also know number of clusters – 2 (the drone touches the land surface or not during landing) and prior to use clustering we were applied FA on the dataset for reducing number of features.

Performance measurement – during the project own implementation of the F1-score, accuracy and recall score were provided. The main reason to write own metrics measurement functionality is time series which represents state of the drone and particularly a touch moment during a landing.

Benchmark

If drone automatic landing detection will work with accuracy equal or above than 0.9 for the f1 score metric, then this project solution is acceptable and useful. If accuracy will be less than desired, then automatic landing will make a lot of False Positive events and drone during the landing will crash from high altitude, that can damage the drone, or drift and may cause an injury (cut, bruise, etc.) to the user, other people or animals.

III. Methodology

Data Preprocessing

During this project, we need to select relevant features for further analysis and data transformations. Also, we need to care about the final solution performance on the drone for the real-time processing of the data. For ICA and FA the dataset was normalized using logarithm function. Since the logarithm function requires values larger than zero, then selected features were scaled to the positive values' range (if in origin they were negative), also, outliers detection was applied on some features using the dataset visual analysis.

Feature - "Status" (drone's state) transformed into the separate binary features: STATUS_flight, STATUS_land and STATUS_take_off.

Linear regression (using DecisionTreeRegressor) was applied to determine relevant features (see Table 1).

Table 1. Relevant features using linear regression

Nr.	Feature	Score value	Nr.	Feature	Score value
1	BAR	0.981	11	MAG_Z	0.972
2	MOTOR1	0.977	12	GYRO_X	0.718
3	MOTOR2	0.983	13	GYRO_Y	0.755
4	MOTOR3	0.979	14	GYRO_Z	0.914
5	MOTOR4	0.983	15	PITCH	0.910
6	LIDAR	0.759	16	ROLL	0.895
7	THRUST	0.948	17	YAW	0.953
8	ACC_X	0.823	18	STATUS_flight	1.000
9	ACC_Y	0.831	19	STATUS_land	1.000
10	ACC_Z	0.871	20	STATUS_take_off	1.000

For the further analysis were used: MOTOR1, MOTOR2, MOTOR3, MOTOR4, THRUST, ACC_X, ACC_Y, ACC_Z, MAG_Z, PITCH and ROLL. All those features selection mostly based on data exploratory visualization and some relevant assumptions from experience or export knowledge, due to the linear regression analysis provides relatively the same results for all features and only several features such a LIDAR, ACC_*, GYRO_* have smallest correlation values, but not significant to say they are not impacted by other features.

The feature BAR not used for further analysis because their values depends on a place where the drone flying. The LIDAR sensor's values are too much noised and we can`t really trust them in near a one meter (due to specification of the sensor). GYRO_* features' values also not used due to pitch, roll and yaw features are more precisely estimated and depends on mentioned features.

Features STATUS_* are needed only for the data separation and results validation but they are not primary used for the data analysis.

All features values scaled to range 0.0 – 255.0.

Alter all the outliers detected using such rules: 'MOTOR1': (40.868, 250.447), 'MOTOR2': (53.579, 264.316), 'MOTOR3': (23.536, 274.104), 'MOTOR4': (40.489, 259.557), 'THRUST': (34.542, 235.443), 'ACC_X': (75.0, 190.0), 'ACC_Y': (40.0, 175.0), 'ACC_Z': (75.0, 220.0), 'MAG_Z': (150.0, None), 'PITCH': (80.0, 245.0), 'ROLL': (25.0, 125.0), where in parenthesis given actual range of values. After all the data were scaled (seed Fig. 8 and 9).

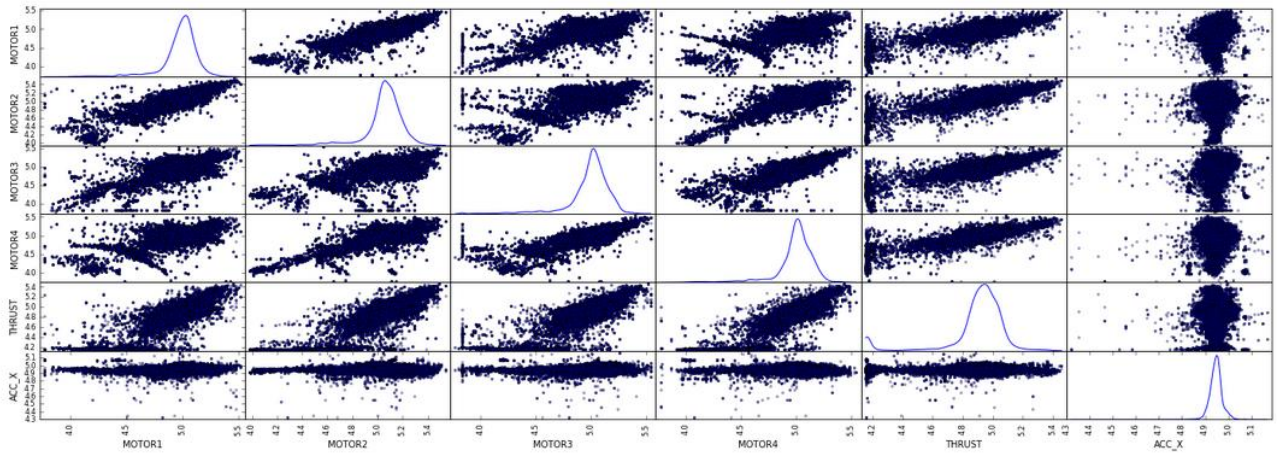

Figure 9. Data with removed outliers. KDE plot for BAR, MOTOR[1-4] and LIDAR

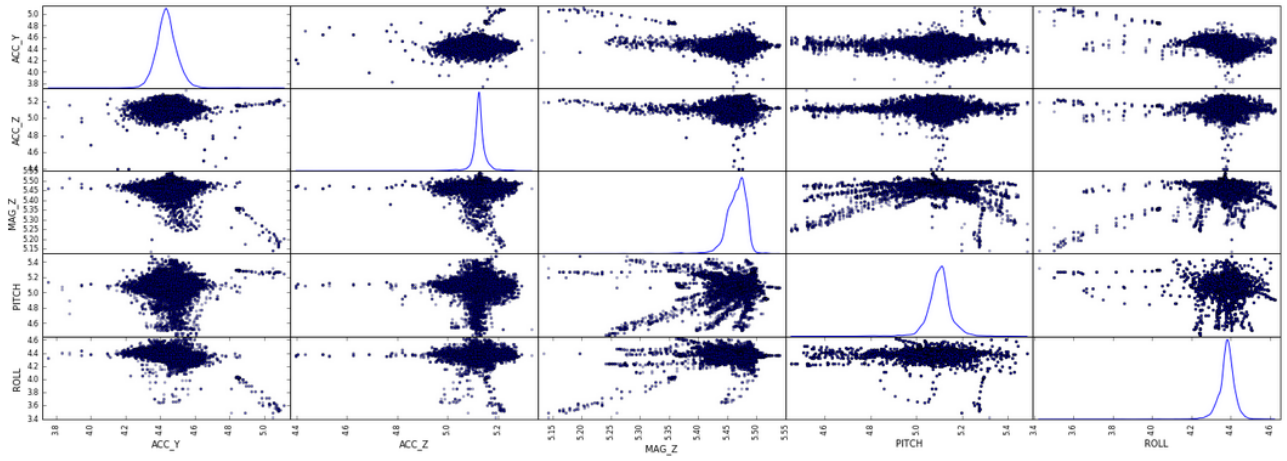

Figure 10. Data with removed outliers. KDE plot for gyroscope, pitch, roll and yaw

FA was applied on all preprocessed data (for drone state – Landing) and only first 5 components were used for further analysis due to their features values has relatively high impact values (see Figure 11 and example of log after FA in Figure 12).

In the Figure 11 in the 1-th component you can see strong correlation between thrust and motors speed. As decreasing of thrust the morors' speed also decreasing. The second and third component has more diverse picture. The ACC_Z values increasing in these components, thus described by only sensor calibration process properties and for better interpreting we need to multiply ACC_Z value with negative one. After all, in the first five components motors' values changes differently, this described by the fact that the Flight Controller always trying to stabilize a drone horizontally during the landing and, for example, wind gusts will raise drone body position changes which should be compensated by motors speed changes. On the other hand other features (pitch, roll, yaw, acc*) changes usually differs before and after the drone touched the land surface.

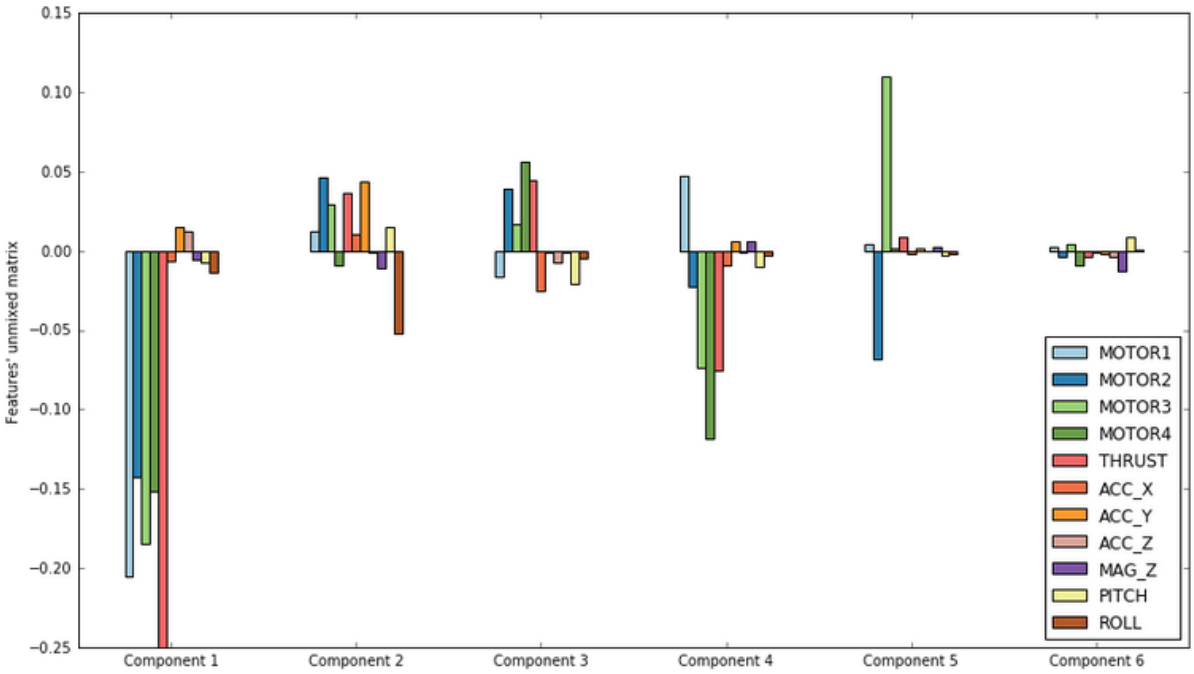

Figure 11. FA results

In the Figure below you can see FA transform result on the data for logID=19. Where the drone touched the land surface in 164.80-165.34 sec.

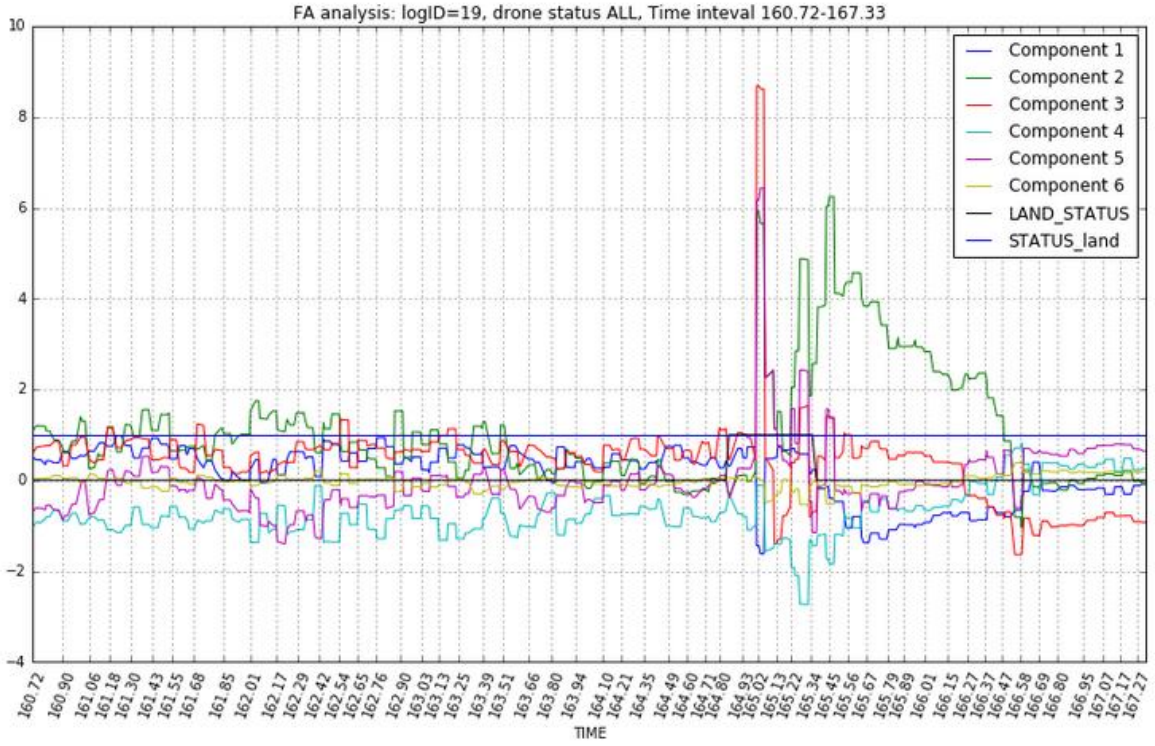

Figure 12. Example: LOGID=19 data representation after applying FA.

Implementation

Performance metrics

As mentioned before for the performance measurement are used: f1 score, precision and recall score. In this project, author has own implementation of the all such metrics. This due to the landing detection may arise during time period (data are time series). There are implementation details (see implementation in the *measure_accuracy*() method) :

a) FP – estimated during the landing, before the drone was touched the surface. Only single FP prediction value necessary to satisfy singular FP. Otherwise singular TN should be satisfied;

b) FN – estimated during the real touch time moment. If no landings were predicted, then singular FN should be satisfied, otherwise only singular TP should be satisfied.

Clustering

K-Means and GMM were applied after FA transform on whole data. The dataset divided into the training (logid - [0, 11]) and testing (logid - [12, 24]) sub sets. In the figure below the clustering results are shown:

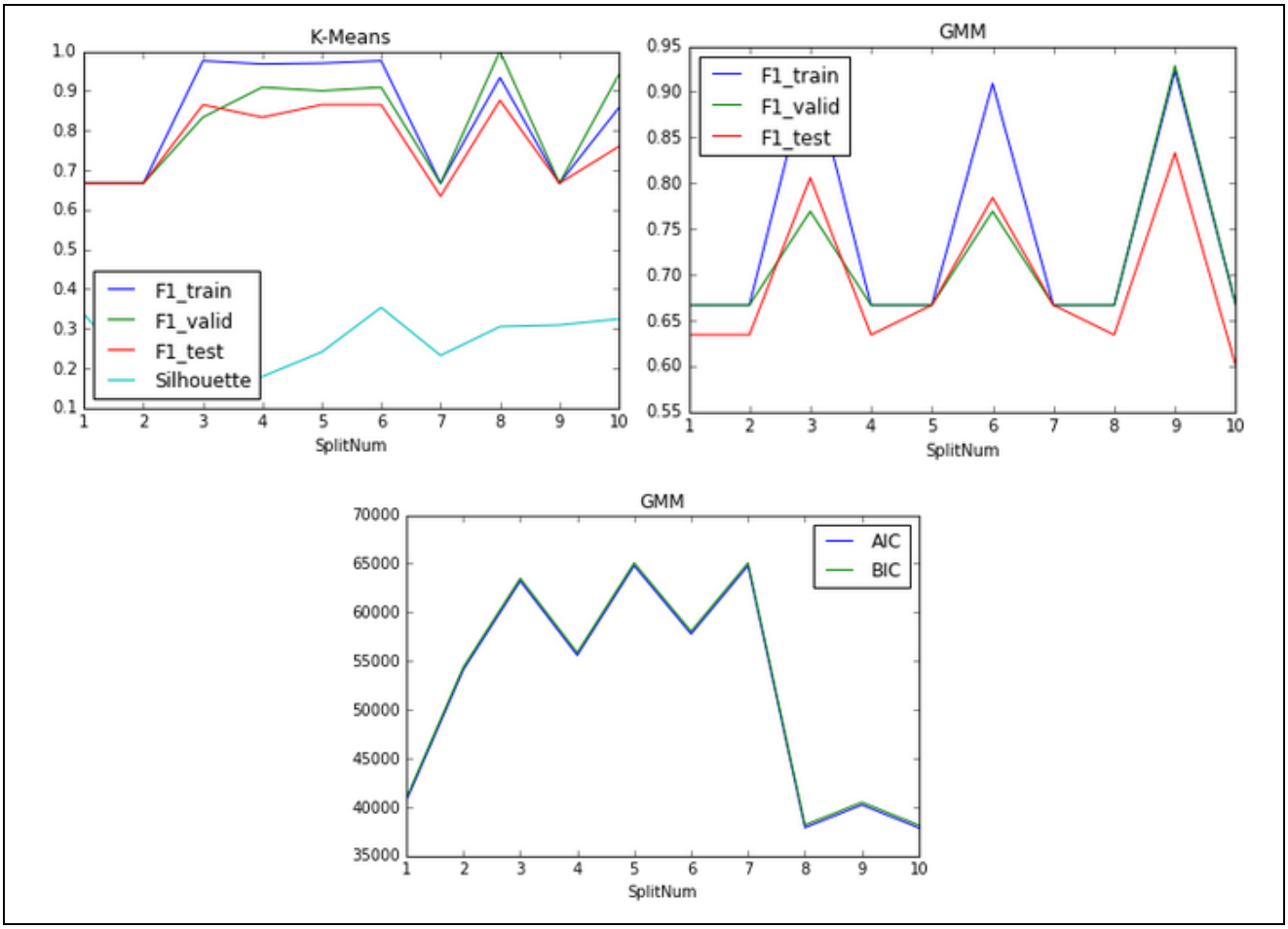

Figure 13. K-Means and GMM clustering results for the 10 splits. Metrics reported: f1_score, recall and precision. For testing used only data from the testing subset.

The best results for K-Means provided in split 9-th (or in 8-th, then starting numbering from 0) by f1 score (0.88) and for GMM in the 9-th split by f1 score (0.83). The visual representation of the clustering results shown below.

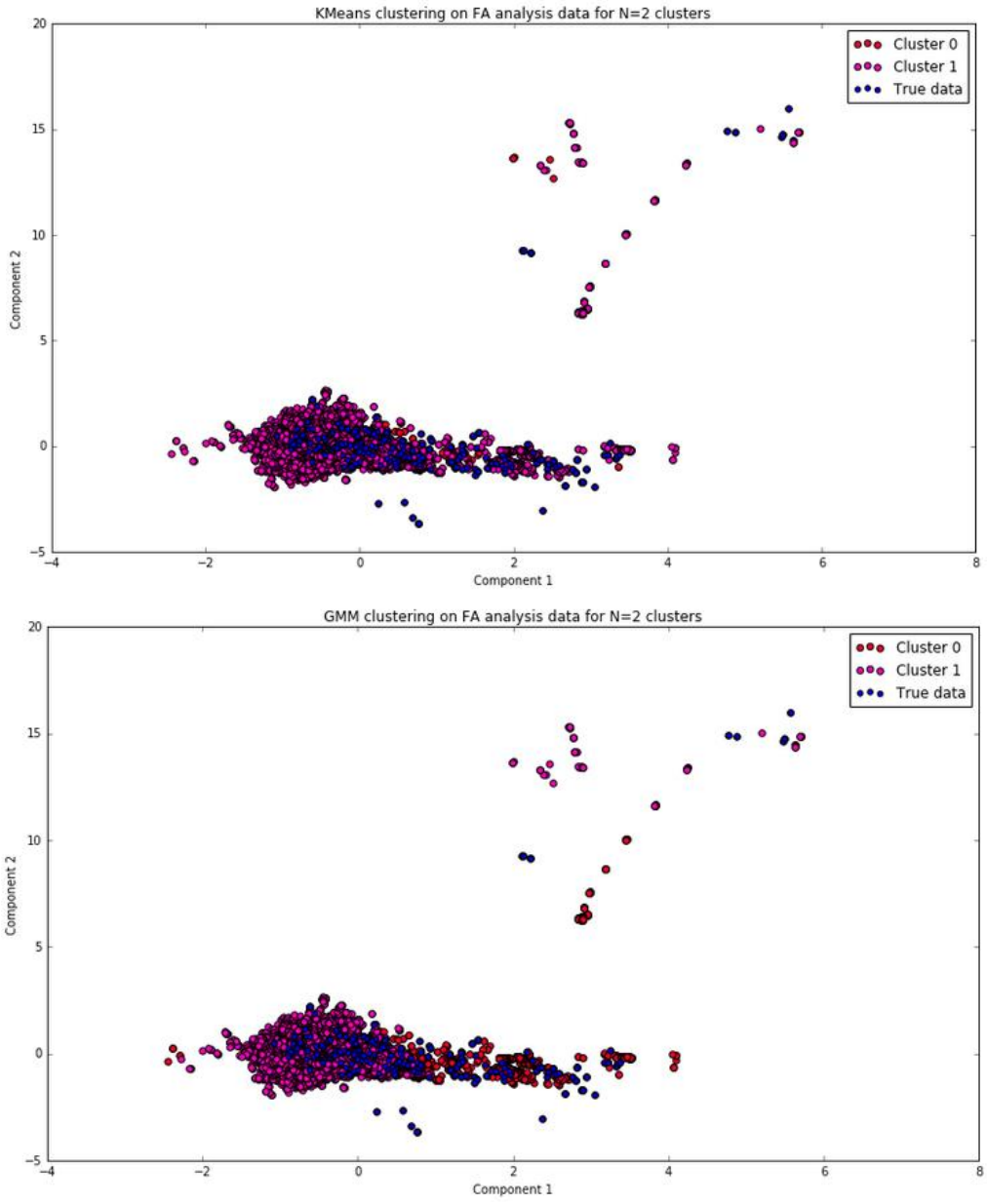

Figure 14. Initial K-Means and GMM clustering results and components 1,2 visualization, , where true data is marked real touching

Refinement

K-Means is one of the GMM types. During refinement covariance type ‘tied’ for GMM was selected by f1 score value which is finally equal to 0.941. This result was obtained during 200 iterations.

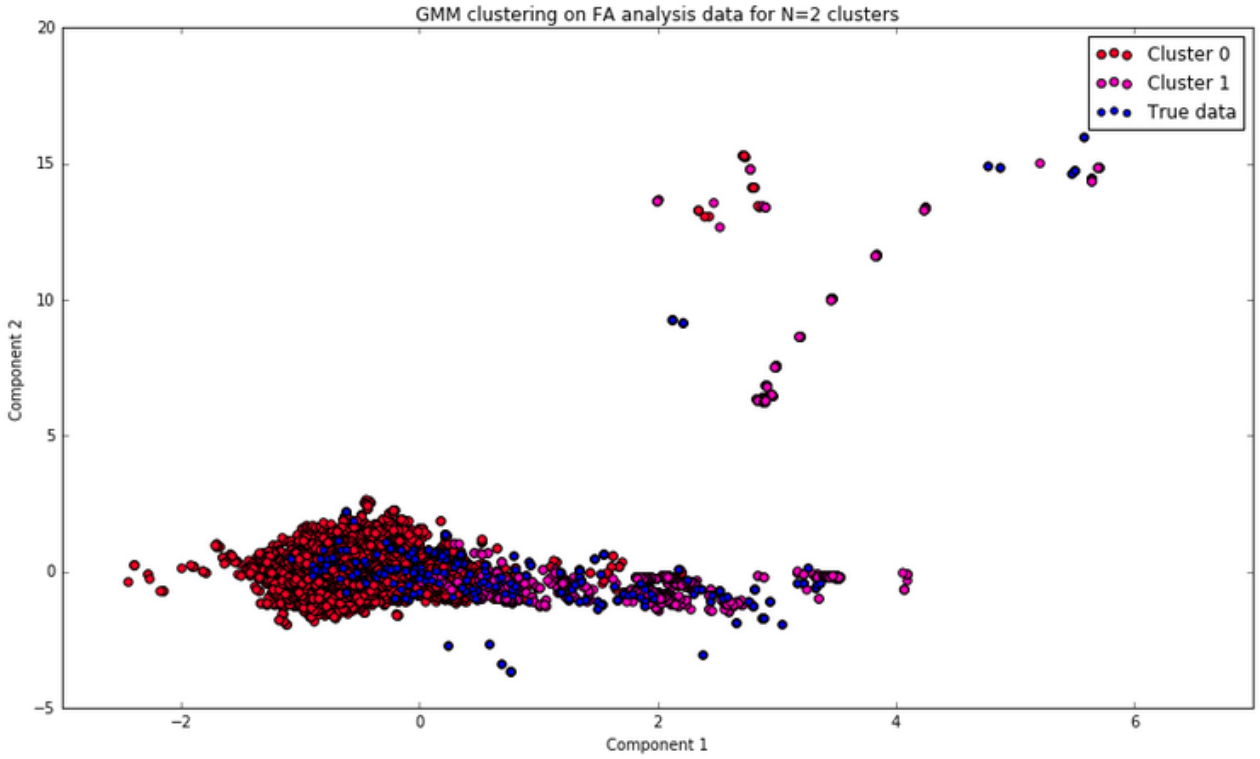

Figure 15. GMM clustering results after refinement, where true data is marked real touching

As you can see in Figure 15, true data smoothed for both clusters, but we need take into account, that such true point also represent drone landings due to data has type of time series.

IV. Results

Model Evaluation and Validation

The final model (using GMM clustering) completely aligning with the project expectations and gives good performance results 0.941 (f1 score) on unseen data within 10 logs with the 12 real drone’s touch moments. The small changes in training data set do not give high impact of results (see Table 2).

Table 2. GMM clustering results during 10 folds for f1 score

Description/ Dataset	**Training dataset, logs - [0, 11]**	**Validation dataset, logs - [0, 11] only 35%**	**Testing dataset, logs [12, 24]**
Mean	0.89	0.84	0.85
Std	0.12	0.09	0.10
Max	0.98	0.92	0.93
Min	0.67	0.67	0.67

Justification

The final results are stronger than benchmark results, due to GMM will trained with ‘tied’ covariance type and K-Means is only a case of GMM.

K-Means clustering is a case GMM, but last gives better results during the final model tuning.

F1 score value 0.941 for unseen data is good and completely fits with benchmark requirements. Also small variation in training data did not significantly impacted the model training results and standard deviation for the testing dataset was only 0.1 of f1 score during 10 folds.

The dataset of this project belongs to time series data type, then clustering techniques are more adequate approach than, for example, SVM or Decision Trees.

V. Conclusion

In the Figure 16 you can see how well the final model works on unseen data. There no False Positives before real touch moment except logid 14, where the touch moment detected a little bit early with ~0.20 sec, but this mistake was not actually significant if we consider that usually drone’s landing speed is low.

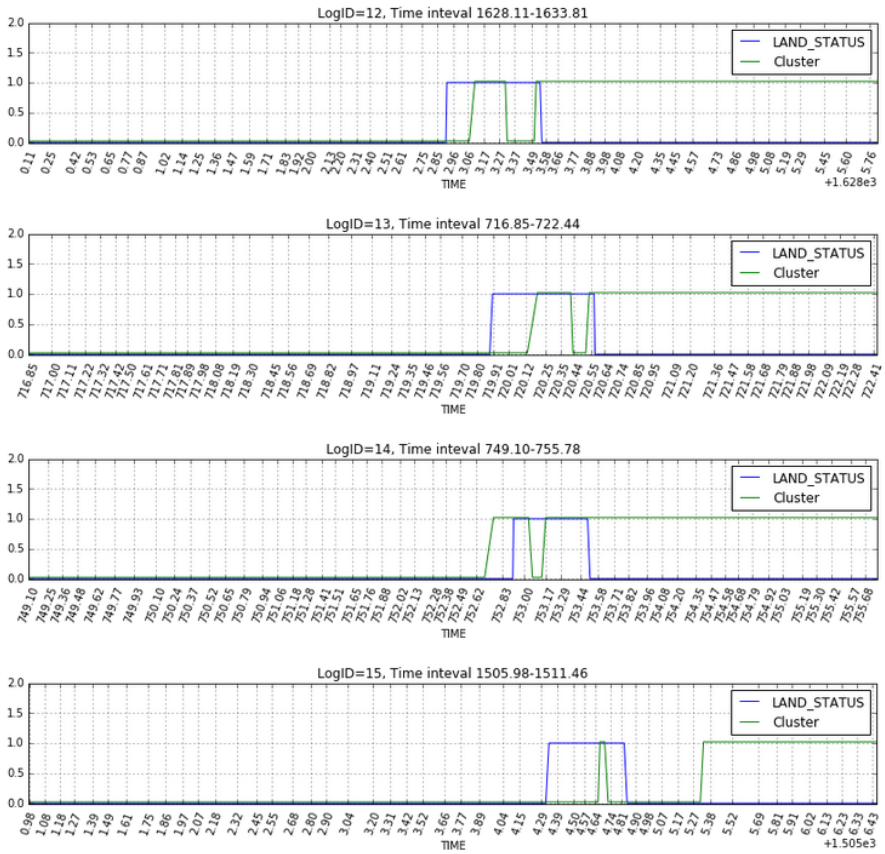

Figure 16. The drone touching moment prediction on unseen data for logs 12, 13, 14 and 15

Reflection

This project's problem is still actual for the drone industry. More and more drone manufactures trying to use machine learning techniques and algorithms to make decisions in real-time and with minimal computation resources. This project is not exception and attempt to solve automatic landing detection problem.

One of the most problematic and interesting aspects of this project is data type – time series, but as we need detect a touch moment we can`t use time series as input data for a prediction due to a decision time is strongly limited and we need detect touching moment with the land surface as soon as possible.

Dataset preparation and preprocessing takes a lot of time, but largest time was spent in modeling task. Different algorithms were tried SVM, Decision Making models, K-Means, GMM (first two were rejected due to bad benchmarks' results). Also in this project accuracy metrics were implemented by author.

The final model fit with the project expectations and benchmark, finally it gives 0.94 f1 score on unseen data. May be critical moment for this project is find trade-off between False Positives and True Positives. The first may arise drone crash and the second will arise drift of the drone, both are with negative sign.

GMM is fast clustering algorithm and scaling training dataset will not significantly impact training time. This gives more opportunities for future improvements.

Improvement

I think that improvements may occur in modeling stage, where training data will have additional features, those allows to do clustering better. Also, largest training and testing dataset will give more robust results.

Techniques that would be interesting to implement in future are time series pattern matching and learning algorithms. If we will use relatively small time window, for example, 0.3 sec for the drone touching moment prediction, then this may be improved current final benchmark value.