Using Machine Learning to Identify Activities of a Flying Drone from Sensor Readings

Roman Barták and **Marta Vomlelová**
Charles University, Faculty of Mathematics and Physics
Malostranské nám. 25, Praha 1, Czech Republic

Abstract

The dawn of autonomous robots brings a question of automated modeling of robot behavior such that the learned robot capabilities can be used to plan robot activities. To bridge the continuous world of sensor readings and control signals with the symbolic world of planning, one needs to identify robot activities as somehow compact behaviors that can be repeated later when a given activity is planned to be performed. In this paper we focus on identifying activities from a sequence of sensor reading and corresponding control signals by using the methods of machine learning, both supervised and unsupervised. The methods are experimentally evaluated using data from a flying drone.

Introduction

Autonomous systems and robots are becoming increasingly interesting for general public as well as for researchers. Flying drones are frequently discussed by media, but these are usually remotely controlled devices with limited autonomy. There are autonomous systems such as self-driving cars (Google 2016) and warehouse robots (Kiva 2016) that are carefully pre-programmed by humans to do a specific task in uncertain and dynamic environments, but these systems cannot do any other task without being re-programmed. In our research we address truly autonomous systems that can learn their capabilities themselves and based on them they can solve, in principle, any task that their hardware allows. Such systems typically require planning capabilities and to do planning, the system needs a formal model of a planning domain (Ghallab et al. 2004). Reinforcement learning (Abbeel et al. 2007) and deep learning show very promising results recently in many areas of autonomous systems but mainly in tasks that can be solved by reactive techniques rather than by deliberative techniques. To do more complex reasoning about the future, some abstract search techniques are still necessary as, for example, demonstrated by the AlphaGo system (Silver et al. 2016). Unfortunately, there is still a big gap between the continuous world of control systems for robots (reactive systems) and the symbolic world of planning and abstract reasoning based frequently on logic and probability theories (deliberative systems).

To bridge the gap, it would be useful to build activity models automatically from observations of agent behavior. Then the activity model can be used for activity planning by providing abstract description of activities and also for plan execution by providing control knowledge for performing individual activities. One of the first steps to have such a system is identifying "homogenous" time periods in the behavior of agent that could be marked as activities.

This paper addresses the problem of identifying primitive activities of a flying drone from sensor readings. By primitive activities we mean activities corresponding directly to commands used to control the drone, such as flying forward or flying backward. By sensor readings we mean information from an inertial measurement unit that integrates accelerometer, gyroscope, and magnetometer, but not yet more complex sensors such as a camera. As the sequence of commands is known together with sensor readings, this gives us opportunities to exploit and compare both supervised and unsupervised learning techniques. In the paper we present several such techniques with their preliminary evaluation.

Used Platform

To do the study, we used an inexpensive flying drone AR.Drone 2.0 by Parrot Inc. The flight is generally controlled by sending pitch and roll angles (relative to a pre-set limit) and vertical and yaw speed (Figure 1). The commands are sent at 30 Hz and the drone's firmware then tries to reach and maintain given values until the next command arrives.

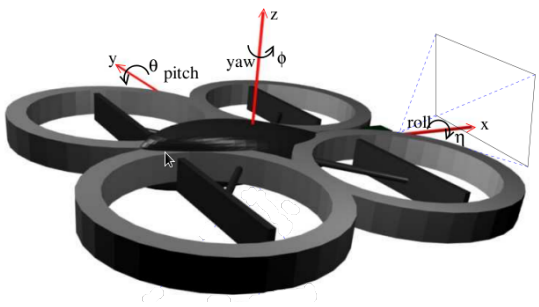

Figure 1: AR.Drone and its coordinate system and angles. (Krajník et al. 2011)

Though the drone is equipped with two cameras, in our study we use only the other sensors, namely a 3-axis gyroscope, a 3-axis accelerometer, and a magnetometer. Altitude is measured using an ultrasound sensor and a pressure sensor, which is used in higher altitudes out of the ultrasound sensor's range. Data from non-visual sensors, so-called *navdata*, are sent from the drone at 15-200 Hz depending on setting and contains in particular roll and pitch angles, azimuth, altitude, and a speed vector in the drone centered coordinate system (Figure 1). The network latency of transmission of those commands and data is approximately 60 ms.

To control the drone we used the YADrone software (YADrone 2016) extended by a module for collecting data with commands sent to the drone and with sensor readings from the drone. Figure 2 shows the simplistic interface of this program. We controlled the drone manually using the keyboard and we collected data for a simple indoor flight – we navigated the drone to fly forward for a few meters and then to fly back to the lift-off location. This was repeated several times, with occasional interruption, when we navigated the drone to fly on one side and to return back with some rotations; all at the same altitude. This way we collected data with rotations, forward, backward, left, and right movements. At any time, we used a single command to achieve intended behaviour, such as forward flight, though sometimes a different command was used to correct the behaviour, for example to correct the direction due to wind from propellers in a narrow corridor.

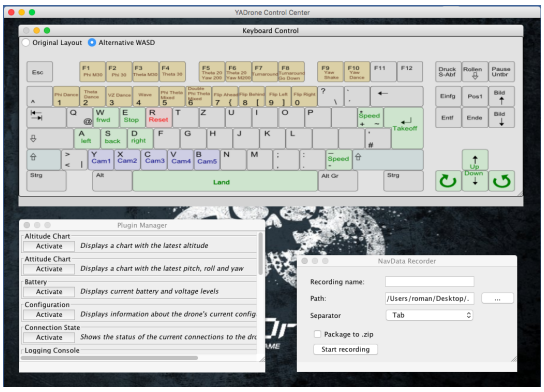

Figure 2: YADrone software with the data grabbing module.

Related Works

Activity recognition is a long-term studied area, but mainly in relation to human activity recognition. For our work it is particularly relevant sensor-based human activity recognition using wearable sensors (Bulling et al. 2014; Attal et al. 2015). There are several notable differences between drone and human activity recognition. First, the set of activities to recognise is obviously different for humans (siting, lying, walking, running etc.) and for drones (flying forward or sidewise, rotating etc.). Second, a wider set of sensors is usually available for drones though the core sensors, namely accelerometer and gyroscope, are used in both areas. Third, sensor data from drones are more noisy (Figure 4) and hence activities are harder to recognise. Finally, control signals for drone movement are directly accessible so manual data annotation is not necessary (for primitive activities), which simplifies application of supervised learning techniques.

In this paper we will adapt methods used for human activity recognition, in particular, those for unsupervised activity recognition by Kwon et al. (Kwon et al. 2014), as well as classical supervised learning techniques.

Overall Concept

We applied a standard machine learning workflow. We developed a software for collecting sensor and control data from AR.Drone that provides two data sets – tables – collected from a drone flight. One table contains time annotated sensor readings; each row describes information from all sensors as provided by the AR.Drone (navdata). The other table contains time annotated commands sent to the drone. We manually pre-processed the data, which includes selection of rows, calculating window aggregations (mean, variance, FFT mean and variance), normalizing values, and joining the tables (for supervised learning). For unsupervised learning we applied and compared clustering approaches k-means, mixture of gaussians (GMM), and hierarchical clustering (HIER) as used in (Kwon et al. 2014) and we also applied a Hidden Markov Model learning that assumes temporal information. Since none of the unsupervised learning methods discovered true control activities in a satisfactory way, we tried a supervised learning technique, namely a decision tree learning, to discover how a given activity depends on values of sensor readings. We have chosen decision trees to keep the models simple and understandable for humans. We also added information from the clustering algorithm (the cluster is added as another attribute to each row) to check if it can improve the decision tree prediction. In the rest of the paper we will describe these steps in detail.

Preprocessing

As we already mentioned, we collected data from a drone flight into two tables. One table with `25986` records described time annotated sensor readings. The first attribute of each record (row) was a computer time (`ctime`) and there were `24` additional attributes (navdata from the AR.Drone). The other table with `2275` records described time annotated command data. Again, the first attribute of each record was a computer time (`ctime`) and there were three additional attributes (`LeftRight`,`FrontBack`,`Angular`) that describe the command data. For example, the numeric value of the `LeftRight` attribute indicates by a sign whether the drone is ordered to fly right, left, or not on any side. Notice also that there are much fewer commands sent to the drone than the number of sensor readings obtained from the drone.

Merging of Tables

In the supervised learning we learn the function $navdata \rightarrow command$ described as a single table, where each sensor reading corresponds to a command. Hence the first decision was how to join the command and navigation data by

the ctime variable. Since the reaction on a command takes time (the command is sent to the drone via WiFi and the sensor reading is sent back to a computer), we determined the best time delay as follows. Based on how the command system in the AR.Drone works, we modified the table with commands. For each sensor reading at time t we introduced the command line at time t such that the command is taken from time t_1, $t_1 \leq t$, in the original table and there was no command at time t_2, $t_1 < t_2 \leq t$. This way we got the command table of the same size as the navdata table (except the first sensor readings that appeared before the first command). We then applied the decision tree learning algorithm (as in the section Supervised Learning) to predict the commands FrontBack and LeftRight for different delays in sensor readings. The delay corresponds to shift of rows when merging the tables. Figure 3 shows the prediction error for different shifts (delays) of rows.

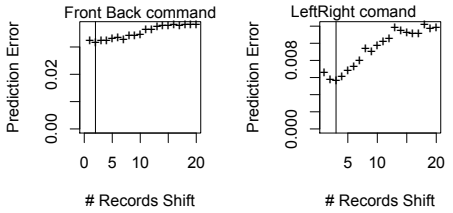

Figure 3: Prediction error based on the shift between navdata and commands for decision tree learning of FrontBack (left) and LeftRight (right) commands.

The minimal error was reached for the shift of 2 or 3 records that corresponds to the difference of 10 or 15 in ctime. All further experiments are done on data merged with 10 time-units delay. From the merged table, we removed all records before the first command arrived. For clustering we also standardised the inputs (we scaled the variables to have mean = 0 and variance = 1).

Windows Size and Aggregation

For unsupervised learning we added some aggregated data. Motivated by (Kwon et al. 2014) we used a window size 40 with 50% overlap between consecutive windows (for comparison we also used window size 1, but the results were not much different as will be presented later). For each window, mean and variance of pitch, roll, yaw was calculated. For window size greater than 1 also Fast Discrete Fourier Transform (R 2016) mean and variance were computed.

In our experiment seven activities were possible obtained from commands FrontBack, LeftRight, Angular with directions (positive or negative) of movement. We encode them using triples of digits representing directions: 1 for negative, 2 for the value around zero, 3 for positive sign (Table 1). For example, 122 means flight forward, while 322 means flight backward (222 means 'do nothing'). To each window we assigned the most common activity in the window. Figure 4 shows the mean values of pitch, roll, and yaw for the most frequent activities as a function of time windows. These data are much more noisy than data for human activity recognition (Kwon et al. 2014), but still there are patterns characterising the activities.

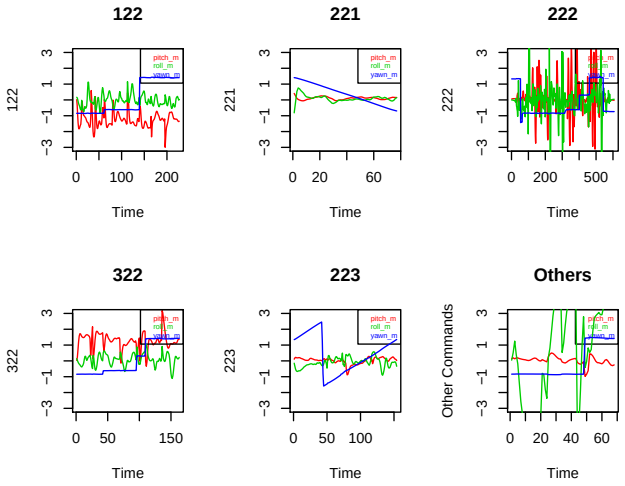

Figure 4: Mean pitch, roll, yaw for windowed data split according the true command. In the first column, we can see lower pitch for the forward movement compared to higher pitch for the backward movement below.

Equal Frequency and Full data

As Table 1 shows, the number of samples for different activities differ significantly, which may influence the clustering algorithm. Hence, we performed experiments both on full data and on equally sized data. To create equally sized samples, we omitted two low–frequency activities 212 and 232 and we selected 50 random records for each remaining activities, i.e. 122, 322, 221, 223, 222.

Activity	FrontBack		LeftRight		Angular		
	fwd 122	bwd 322	left 212	right 232	cw 221	acw 223	nothing 222
#samples	227	163	21	47	77	154	607

Table 1: Encoding of assumed activities and the numbers of window samples for different activities.

Unsupervised Learning

As already mentioned, we tried four different methods of unsupervised learning: *hierarchical clustering*, *k-means clustering*, *gaussian mixture model*, and *Hidden Markov Models* (HMM). While the first three methods do not consider time annotation, HMM is a model of the system evolution in time. As we will see, these methods produce different results. The methods were applied to windowed data, where each record corresponds to one window, and the attributes of the window are mean and variance values of pitch, roll, yaw and Fast Discrete Fourier Transform mean and variance.

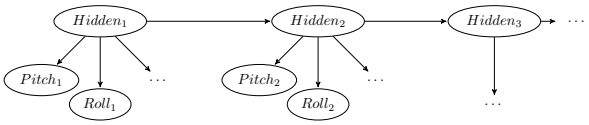

Figure 5: A (simplified) HMM model. Each time is represented by one instance of the hidden variable with each sensor reading as a dependent variable. There are more than tree sensor readings in each time, all taken as conditionally independent given the hidden variable.

HIER, k-means, GMM Clustering

We used hierarchical clustering (HIER) with average–linkage method (James et al. 2013). The cluster tree was cut on the level providing best accuracy.

The k-means algorithm (James et al. 2013) searches k centres in the feature space such that the overall distance of data points from its closest center is minimised.

Gaussian Mixture Model (GMM) can be viewed as HMM (see Figure 5) collapsed in time. GMM has one hidden variable representing the class. The attributes are assumed to be independent given the class. For each class value, the distribution of any attribute is assumed to be gaussian. We used the package `mclust` (Fraley et al. 2012).

For each algorithm we tried the number of clusters between 2 and 15 and we selected the number of clusters with the maximal accuracy (Table 2). This number was different for different methods and it was never higher than 8. We did not experiment with automatic cluster number selection.

Hidden Markov Model

For sensor readings, we may expect some dependence of two subsequent sensor readings. To model this dependence we learned a Hidden Markov Model using the function `learn.hmm()` from the library `depmixS4` (Visser and Speekenbring 2010). A HMM aggregates the state of the robot in an unobserved, hidden state. The sensor readings depend probabilistically on this state and there is also a probabilistic transition from the current state to the next one. A simplified HMM is shown in Figure 5.

By learning HMM, we learn the maximum likelihood (ML) model for each sensor $p(Sensor_k|Hidden)$ and the ML state transition matrix $P(Hidden_i|Hidden_{i-1})$. As a side-effect, we know the most probable hidden state for each time. We take the hidden states as cluster indicators.

Experiment Results

As in (Kwon et al. 2014) we use two evaluation metrics to measure the quality of clustering: accuracy and Normalized Mutual Information.

The evaluation of accuracy was adjusted as follows. To each activity, we assigned a cluster that has the largest portion of the true cluster (the true cluster is formed by the records for that activity). If two or more activities share the same cluster, then the cluster was assigned to the activity that has more records in that cluster, and for the other activity the second largest cluster was picked. Notice that this definition is rather strict. It does not allow for one activity to be represented by more clusters. Therefore, the optimal number of clusters tends to be small.

As another measure, we used Normalized Mutual Information (NMI) ranging from 0 – no information between distributions to 1 – full agreement (Strehl and Ghosh 2002). The NMI is defined as follows:

$$NMI = \frac{2 \cdot \sum_{i=1}^{r} \sum_{j=1}^{s} n_{i,j} \log\left(\frac{n \cdot n_{i,j}}{n_i \cdot n_j}\right)}{\sqrt{\sum_{i=1}^{r} n_i \log \frac{n_i}{n} \sum_{j=1}^{s} n_j \log \frac{n_j}{n}}}$$

where r is the number of clusters, s is the number of classes (different activities), $n_{i,j}$ is the number of records in cluster i and class j and n_i, n_j, n are marginals corresponding to clusters, classes, and overall marginal.

For each clustering method (HIER, k–means, GMM, HMM) we evaluated three experimental settings and measured the accuracy and NMI (Table 2). The experiments differ in the window size (either window size 40 with 50% overlap or window size 1, and in the data used (either the equal–frequency subsample or the full data).

method	HIER	k–means	GMM	HMM
accuracy				
w=1, eq.f.	44.8%	43.6%	36.0%	**48.8**%
w=40, eq.f.	42.8%	38.0%	**49.6**%	44.4%
w=40, full	44.8%	42.8%	47.2%	**48.0**%
NMI				
w=1, eq.f.	0.326	0.235	0.202	**0.411**
w=40, eq.f.	0.290	0.230	0.370	**0.422**
w=40, full	0.325	0.176	**0.392**	0.350
# clusters				
w=1, eq.f.	6	6	6	6
w=40, eq.f.	6	4	5	8
w=40, full	7	5	6	5

Table 2: Accuracy and NMI for clustering methods (HIER, k–means, GMM, HMM) with the window size (w) 1 or 40, and full or equal-frequency data. In each experiment, the number of clusters maximizing accuracy was selected (the third part of the table) and NMI was calculated for the same number of clusters.

The accuracy of random guessing is 20% since we have 5 distinct classes (after removing two less-frequent activities). Hierarchical clustering and k–means provide slightly worse results than GMM and HMM clusterings, but even the best results for drone activity recognition are much worse than for human activity recognition, where GMM reached value 1 both for accuracy and NMI (Kwon et al. 2014). The reason could be that data are much more noisy for AR.Drone (see Figure 4). This may not be the problem of sensor readings rather than of the less stable flight. However, this requires further research. Note also that putting equality between activities and commands may not be the right thing as, for example, flying forward requires more commands in unstable (such as windy) environments.

Comparison of Obtained Clusters

Interestingly, the split of records into clusters is different for different methods of clustering. Table 3 shows normalized mutual information (NMI) between clusterings. The clusters obtained by HIER and k-means are most similar (NMI=0.69), HMM clusters are most different from these (NMI=0.24 and 0.28). Hence different clustering methods use different information to build the clusters, which may be exploited for example in decision tree learning based on clusters. The difference of clusterings is illustrated in the projection to two attributes in Figure 6.

method	HIER	k-means	GMM	HMM
HIER	1	0.69	0.38	0.24
k-means		1	0.43	0.28
GMM			1	0.36

Table 3: NMI comparison of different clustering methods.

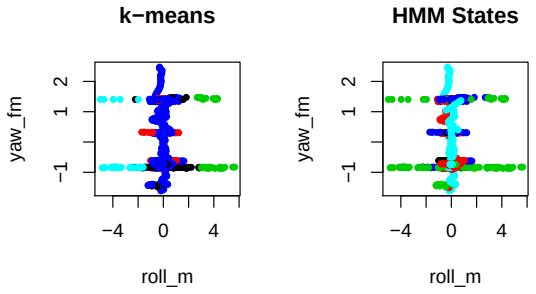

Figure 6: Projection of the clustering results for k-means (left) and HMM (right) into two attributes (yaw and roll).

Supervised Learning

Since the clustering algorithms were far from perfect representation of activities, we tried to learn a sensor description corresponding to each activity by techniques of supervised learning. In particular, we will learn the function $navdata \rightarrow activity$ using the decision trees.

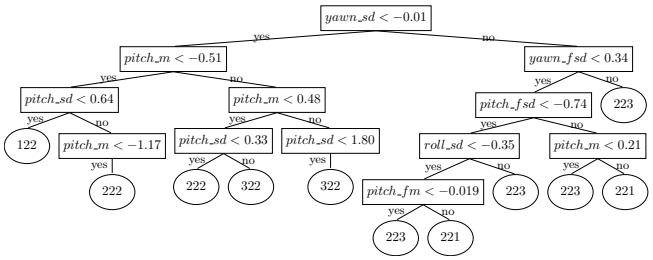

Figure 7: The decision tree predicting the activity based on the windowed input attributes.

Decision Trees

As a supervised learning method we used classical decision tree learning using the library `tree` (Ripley 2016). We performed 10 times random selection of stratified data, 50 training examples from each class, 5 test samples from each class. In each run, we constructed the decision tree and we used cross validation on the training data sample to select best pruning of the tree. Then, we evaluated this pruned tree by the test data. Figure 7 shows the pruned decision tree for the `activity` based on the windowed data.

Decision Trees Exploiting Clusters

We were also interested in whether clustering helps to identify the activity. This has been implemented by using clusters assigned by each unsupervised learning method (HIER, k-means, GMM, HMM) as attributes of records. Only these cluster attributes were then used for decision tree learning. Figure 8 shows the obtained decision tree for the `activity` based on the cluster assignment.

Model Comparison

We compared the model accuracy for decision trees constructed from different input data. The decision tree based on clustering results (see the previous section) has the accuracy 65%. This means that the clustering contains quite high potential to discover the true activity. The prediction based on windowed data was even better (81.2%) and adding the cluster attributes improved the accuracy only very little. Table 4 summarises the results of accuracy with respect to the test data. In any case, the decision-tree learning method achieved much better results than any clustering method.

clusters	clusters	data	both
5	0.648	0.812	0.844
6	0.760	0.800	0.828

Table 4: Accuracy comparison for a decision tree based on cluster assignment, on the windowed data, and on the combination of both. All numbers are averaged over 10 experiments. The first row gives results for 5 clusters, the second for 6 clusters.

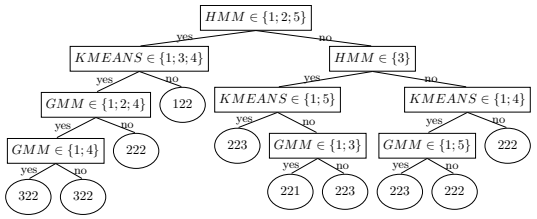

Figure 8: The decision tree predicting the activity based on the cluster assignment by the algorithms HIER, k-means, GMM, and HMM.

Conclusions and Future Research

In this paper we studied two approaches to identify activities of a flying drone, one based on unsupervised learning (clustering) and one based on supervised learning (decision trees). It seems that identifying blocks of time, that could be called activities, is relatively straightforward, if primitive moving activities are assumed only. We have found the obtained clusters reasonable enough (see Figures 6 and 8), but as the empirical evaluation using the command data showed, the unsupervised learning provided sensor reading clusters that do not correspond fully to the original commands and accuracy was much worse than for human activity recognition. The decision tree learning seems much more accurate than clustering regarding primitive activities – control commands. Also the experimental results showed the accuracy of decision tree learning can be slightly improved by including information about clusters found by the unsupervised learning methods.

There are several directions for future research. It is possible to exploit information from cameras, such as optical flow, as additional sensor input. Another possible future step is application of the presented techniques, with the focus on unsupervised learning, to identify more complex moving patterns that repeat when drones perform some mission – for example circular or lawn-mower patterns when searching for an object (Bernardini et al. 2014). These movements do not correspond to a single command and hence are more interesting from the perspective of planning using more complex activities. Having activities comprising more commands brings another interesting problem, namely learning how to perform such an activity (how to control the drone to execute the activity). Reinforcement learning seems to be a good and verified technology there (Abbeel et al. 2007). From the perspective of activity planning it would be also interesting to learn some abstract description of such complex activities in the form of activity preconditions and effects (Ghallab et al. 2004). This is critical for building planning domain models that can be used to obtain even more complex goal-driven behavior of autonomous systems.

Acknowledgments

The research is supported by the Czech Science Foundation under the project P103-15-19877S. The authors would like to thank Tomáš Procházka for developing the data grabbing extension for the YADrone software.

References

Abbeel, P., Coates, A., Quigley, M., and Ng, A.Y. 2007. An Application of Reinforcement Learning to Aerobatic Helicopter Flight. In *Advances in Neural Information Processing Systems (NIPS)* 19, MIT Press.

Amazon Robotics (former Kiva Systems), 2016. `https://www.amazonrobotics.com/`, Accessed August 22, 2016.

Attal, F., Mohammed, S., Dedabrishvili, M., Chamroukhi, F., Oukhellou, L., and Amirat,Y. 2015. Physical Human Activity Recognition Using Wearable Sensors. *Sensors* 15: 31314–31338.

Bernardini, S., Fox, M., and Long, D. 2014. Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions. In *Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling (ICAPS)*, 445–453. AAAI Press.

Bulling, A., Blanke, U., and Schiele, B. 2014. A Tutorial on Human Activity Recognition Using Body-worn Inertial Sensors. *ACM Comput. Surv.* 46(3): 33:1–33:33. ACM.

Fraley, Ch., Raftery, A.E., Murphy, T.B., and Scrucca, L., 2012. mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation, Technical Report No. 597, Department of Statistics, University of Washington.

Ghallab, M., Nau, D., and Traverso, P., 2004. *Automated Planning: Theory and Practice.* Morgan Kaufmann Publishers.

Google Self-Driving Car Project, 2016. `https://www.google.com/selfdrivingcar/`, Accessed August 22, 2016.

Hastie, T., Tibshirani, R., and Friedman, J. 2009. *The Elements of Statistical Learning.* Springer.

James, G., Witten, D., Hastie, T., and Tibshirani, R. 2013. *An Introduction to Statistical Learning.* Springer.

Krajník, T., Vonásek, V., Fišer, D., and Faigl, J. 2011. Ardrone as a platform for robotic research and education. In *Research and Education in Robotics-EUROBOT 2011*, 172–186. Springer.

Kwon, Y., Kang, K., and Bae, Ch. 2014. Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications* 41(14): 6067 – 6074.

R Core Team. 2016. R: A Language and Environment for Statistical Computing. `https://www.R-project.org/`.

Ripley, B. 2016. tree: Classification and Regression Trees. R package version 1.0-37. `https://CRAN.R-project.org/package=tree`

Silver, D., Huang, A., Maddison, Ch.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529: 484–489.

Strehl A., and Ghosh, J. 2002. Cluster ensembles a knowledge reuse framework for combining multiple partitions. *Journal of Machine Learning Research* 3: 583–617.

Visser, I., and Speekenbrink, M. 2010. depmixS4: An R Package for Hidden Markov Models. *Journal of Statistical Software* 36(7): 1-21. `http://www.jstatsoft.org/v36/i07/`.

YADrone – an open framework for controlling the AR.Drone 2. 2016. `https://vsis-www.informatik.uni-hamburg.de/oldServer/teaching//projects/yadrone/`, Accessed August 22, 2016.