CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

Master Thesis

Tomáš Nouza

Safe adaptive traversability learning for mobile robots

Department of Cybernetics

Supervisor: **Ing. Michal Reinštein, Ph.D.**

Prague, 2014

Prohlášení autora práce

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne . . . .
Podpis autora práce

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Cybernetics

DIPLOMA THESIS ASSIGNMENT

Student: Bc. Tomáš N o u z a

Study programme: Cybernetics and Robotics

Specialisation: Robotics

Title of Diploma Thesis: Safe Adaptive Traversability Learning for Mobile Robots

Guidelines:

The aim of this work is to propose, implement and experimentally evaluate a control algorithm exploiting machine learning methods, which will enable the robot to *safely* and *autonomously* traverse obstacles in natural environment. The main contribution of this work lies in development and evaluation of a model for predicting the safety of robot states and actions while interacting with the environment. For this purpose it is essential to use data from the following sensors: *inertial measurement unit* Xsens MTi-G, robot *odometry*, 3D *laser scans* measured using rotational SICK LMS-151 and *visual data* from Ladybug3 omnidirectional camera. All these sensors are mounted on a tracked robot developed as part of the NIFTi project (http://www.nifti.eu/). In the first stage, state-of-the-art analysis will be conducted. In the second stage, the proposed algorithm will be implemented for Robot Operating System (ROS, http://www.ros.org/wiki/) (Python or C++). For the purpose of machine learning and testing of proposed algorithms, use of the MATLAB is recommended. A standard dataset for ROS will also be created and released for the robotic community.

Bibliography/Sources:

[1] Kober, J.; Bagnell, J. A.; Peters, J.: "Reinforcement learning in robotics: A survey." The International Journal of Robotics Research 32.11 (2013): 1238-1274.
[2] Thrun, S.; Burgard, W.; Fox, D.: Probabilistic robotics, 2005.
[3] Peters, J.: Policy Gradient Toolbox for MATLAB at http://www.ias.tu-darmstadt.de/Research/PolicyGradientToolbox
[4] Pieter Abbeel, UC Berkley, CS1882013 Artificial Intelligence course http://www.youtube.com/user/CS188Spring2013/videos

Diploma Thesis Supervisor: Ing. Michal Reinštein, Ph.D.

Valid until: the end of the summer semester of academic year 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
Head of Department

prof. Ing. Pavel Ripka, CSc.
Dean

Prague, January 10, 2014

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

Student:	Bc. Tomáš N o u z a
Studijní program:	Kybernetika a robotika (magisterský)
Obor:	Robotika
Název tématu:	Učení bezpečného přizpůsobivého průchodu terénem pro mobilní roboty

Pokyny pro vypracování:

Cílem práce je navrhnout, implementovat a experimentálně ověřit řídící algoritmus využívající metod strojového učení, který umožní mobilnímu robotu bezpečně autonomně překonávat překážky v přirozeném venkovním prostředí. Hlavním přínosem práce bude návrh a ověření modelu robota umožňující predikci bezpečných stavů a akcí při interakci robota s prostředím. K tomuto účelu lze využít dat z následujících senzorů: inerciální měřící jednotka Xsens MTi-G, odometrie robota, 3D laserové skeny měřené pomocí rotujícího SICK LMS-151 a obrazová data ze všesměrové kamery Ladybug3. Všechny jmenované senzory jsou umístěny na pásovém robotu vyvinutém v rámci projektu NIFTi (http://www.nifti.eu/). V první fázi se řešení bude zabývat analýzou stavu poznání. V druhé fázi bude implementován algoritmus pro Robot Operating System (ROS, http://www.ros.org/wiki/) (jazyk Python nebo C++). K učení a testování navržených algoritmů je doporučeno využít prostředí MATLAB. Součástí diplomové práce bude tvorba standardizovaného datasetu pro ROS, který bude po dokončení práce uvolněn pro robotickou komunitu.

Seznam odborné literatury:

[1] Kober, J.; Bagnell, J. A.; Peters, J.: "Reinforcement learning in robotics: A survey." The International Journal of Robotics Research 32.11 (2013): 1238-1274.
[2] Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics, 2005.
[3] Peters, J.: Policy Gradient Toolbox for MATLAB at http://www.ias.tu-darmstadt.de/Research/PolicyGradientToolbox
[4] Pieter Abbeel, UC Berkley, CS1882013 Artificial Intelligence course http://www.youtube.com/user/CS188Spring2013/videos

Vedoucí diplomové práce: Ing. Michal Reinštein, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic	prof. Ing. Pavel Ripka, CSc.
vedoucí katedry	**děkan**

V Praze dne 10. 1. 2014

Abstract

In mobile robotics it is necessary to predict a robot pose on a terrain to guarantee its stability when traversing an obstacle. Usual methods are based on an exact simulation of a robot-surface interaction, but this requires a precise physical model, which can be hard to solve or can be too much complex. The aim of this thesis is to propose and experimentally evaluate an algorithm, based on machine learning methods, which predicts attitude of the robot (roll and pitch angles) in natural environment. The main contribution of this work lies in development and evaluation of models, which can be used for predicting the safety of robot states and actions while interacting with the environment. Three models based on different multidimensional regression methods (linear, piecewise constant and Gaussian process) were trained and compared. As a part of this work, testing dataset was created and will be relased for the robotic community.

Abstrakt

V mobilní robotice je nezbytné předpovídat postoj robota vůči terénu, aby se zaručila jeho stabilita při překonávání překážek. Běžně používané metody jsou založeny na přesné simulaci interakce mezi robotem a povrchem, ale vyžadují precizní fyzikální modely, které může být těžké spočítat, nebo mohou být příliš komplexní. Cílem této práce je navrhnout a experimentálně ověřit algoritmus založený na metodách strojového učení, který bude předpovídat polohové úhly robota (náklon a sklon) v přirozeném prostředí. Hlavním přínosem této práce je vytvoření a ověření modelů, které mohou být požity k predikci bezpečných stavů a akcí při interakci robota s prostředím. Úspěšně byly naučeny a porovnány tři modely založené na rozdílných vícerozměrných regresních metodách (lineární, po částech konstantní a pomocí Gausovských procesů). Součástí této práce bylo rovněž vytvoření testovacího datasetu, který bude uvolněn pro robotickou komunitu.

Acknowledgements

I would gladly like to thank my supervisor Ing. Michal Reinštein, Ph.D. for his knowledge support and permanent flogging me to labour. I also want to express special thanks to Ing. Karel Zimmermann, Ph.D., for writing the skeleton of regression pipeline used in this work, and to Ing. Petr Zuzánek, for its extension and many datasets used in this work. I would also like to thank my family and girlfriend for their patience, soothing and support.

Contents

1 Introduction

1.1 Motivation

In the recent years, lot of work was made in the field of reinforcement learning (RL) for robots. Inspiration sources from nature, where babies come to the world equipped only with a few reflexes and learn the rest of their behavior, like walking, from their interaction with an environment. Every time they fall they remember what was wrong and try to beware of doing that the next time, but if everything goes right, they try to repeat it and optimize their movements for better efficiency. This is, in short, the principle of the reinforcement learning. For right actions (e.g. getting closer to the end-state), an agent gets positive reward, for wrong actions (e.g. obstacle hit, wheel slip, etc.), the agent gets negative reward. Simply, the strategy of the agent is solving an optimization problem which maximizes its reward.

The ideal application of the RL for a robot can be imagined as an arena in which this robot is placed with specified goals and reward rules. The robot randomly tries various actions and gets rewards for them. Progressively it learns how to interact with the environment efficiently.

In the real applications, there is no problem with situations that should be granted with the positive rewards, because everything goes right. In the opposite situations, which are granted with the negative rewards, the robot could be in danger and because no damage during the training phase is allowed, it is not possible to teach the robot these situations (it must not fall into a hole and destroy itself).

Motivation of this work is to guarantee safety for mobile robots by predicting their attitude (pitch, roll) in the future when traversing an obstacle. It is a task of predicting proprioception from exteroceptive data which is, practically, a difficult problem. However, it partially solves the problem of negative rewards in the RL training phase by not allowing the robot to move into the potentially dangerous places where the robot could overturn itself (the pitch and roll angles must not exceed desired thresholds). Presented approach should be used independently on the robot platform, especially in situations where the conventional numerical approach for computing a robot contact points with the surface is difficult. Final implementation will be used as a module for the autonomous adaptive traversability (AT) learning which is currently being developed to extend the training dataset of the state-of-the-art AT method [1] based on the RL principle.

1.2 State of the Art

Although the safe traversability of the mobile robots on a flat surface was solved [2, 3, 4, 5], on a rough terrain it is still a hot topic. Unfortunately most of the natural (forest, cave, etc.) or disaster environments, like collapsed buildings, are unstructured. After the Fukushima disaster in 2011 a lot of attention was payed to the robotic projects focusing on the urban search and rescue (USAR) which led to an increase of funding in this field and to projects like NIFTi [6], TRADR [7], DRC [8] and others.

The traversability on an uneven surface is closely related to the robot platform attitude which is associated with a vehicle stability. Classical approach for predicting the robot attitude and configuration uses analytical solution to compute the contact point of the robot with a surface represented by the DEM (digital elevation map) [9].

The Demo III experimental Unmanned Vehicle project [10] presented a four-wheel platform, which was able to autonomously avoid obstacles and traverse in highly unstructured outdoor environments, tested by the U.S. Army. Among others it computes support surface (even in high grass) from a stereovision system combined with a LADAR data for computing the pitch and roll angles by placing the vehicle mask on this surface.

Tharok *et al.* [11] proposed a general approach for kinematics modeling of *n*-wheel articulated rovers which is used for current Mars rovers. Weak part of this method is that it relies on a perfect knowledge of the underlying terrain which is always uncertain because of a sensor uncertainty. Ho *et al.* extended this approach using the Gaussian Processes (GP) to also predict the traversability in places, where sensors do not provide enough data (due to a sensor occlusion, etc.) and which were previously marked as untraversable for the lack of gaps in the terrain data [12]. GP can handle sensor uncertainties and include them into the traversability prediction. In the following work the authors have improved their approach using GP regression for the terrain features to predict the traversability on an unstable terrain, where the rover-terrain interaction causes deformation of the terrain [13].

1.3 Proposed Method

To predict the attitude of a rover, the *near to far* approach is used. In learning phase, any 3D sensor (like lidar, RGB-D camera or stereo vision system) observes the terrain ahead of the robot and constructs a DEM from it (see the section 3.1). As the robot moves forward, the previously measured data in the form of DEM are slides under the robot body and the new data from the 3D sensor update the DEM in its frontal area. Later on, when the robot is on the DEM filled with known data (composed from multiple measurements), its attitude (position angles of the center of mass) obtained from the inertial measurement unit (IMU) is associated with corresponding DEM features (extracted from this DEM (see the section 3.2) in a similar way as in the image processing), which together forms the training pair (see the section 4.1.1). Pairs obtained in this way are processed and used as samples for the machine learning (ML). Three ML methods are used: the linear regression, piecewise constant regression and the Gaussian process regression.

These trained regression functions are then used for the prediction in testing phase, where the DEM from a desired distance in front of the robot is used as an input, and the robot attitude for that distance is computed (see the section 3.3). This attitude is then compared with the ground truth data obtained from IMU in future time samples, where the robot has moved its center of mass to the desired distance. The root mean square (RMS) error in the testing data is then computed for every ML method and they are compared.

Training and testing dataset took place in both, outdoor and indoor environment. Indoor environment consisted of stairs and corridors, outdoor environment was formed from various palettes formations. The rover crossed over these obstacles several times in different modes operated by a skilled operator. For better description of these environments see the section 5.1.

2 Resources

In order to provide the reader some important background information, this chapter briefly describes the robotic platform and software used in this work.

2.1 Hardware

Fig. 2.1 shows an Unmanned Ground Vehicle (UGV) [14] designed by BlueBotics[1], originally developed for the NIFTi [6] project and is currently used in the TRADR project [7]. The robot has two main tracks and four subtracks (called flippers) for better stabilization so it can traverse over various obstacles (e.g. stair climbing). For a better imagination of the robot size, Fig. 2.2 shows dimensioned blueprint of the robotic platform.

The robot has following sensors: rotating SICK LMS-151 laser scanner (which can provide a 3D point cloud composed of rotated planar scans), Point Grey Ladybug 3 omnicamera and X-sens MTI-G inertial measurement unit (IMU) with a GPS module (placed in the robot center of mass). It can be also equipped with 4-DOF (Degrees of Freedom) robotic arm with mounted ASUS Xtion PRO RGB-D camera and the thermoIMAGER TIM160 infrared camera. Finally, every motor has a position encoder for odometry measurement, as well for a detection of the actual robot configuration. The force in each flipper is measured individually from currents in those motors. Combination of all of these sensors provides a sufficient position and orientation accuracy of the robot.

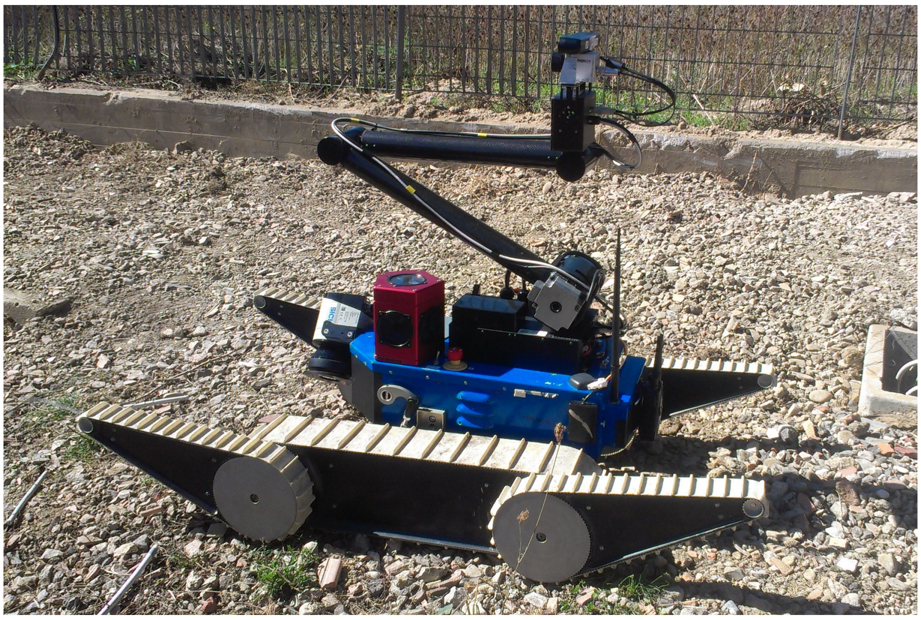


Figure 2.1: BlueBotics UGV [14] used in the TRADR project [7]

[1]Swiss robotics company http://www.bluebotics.ch/

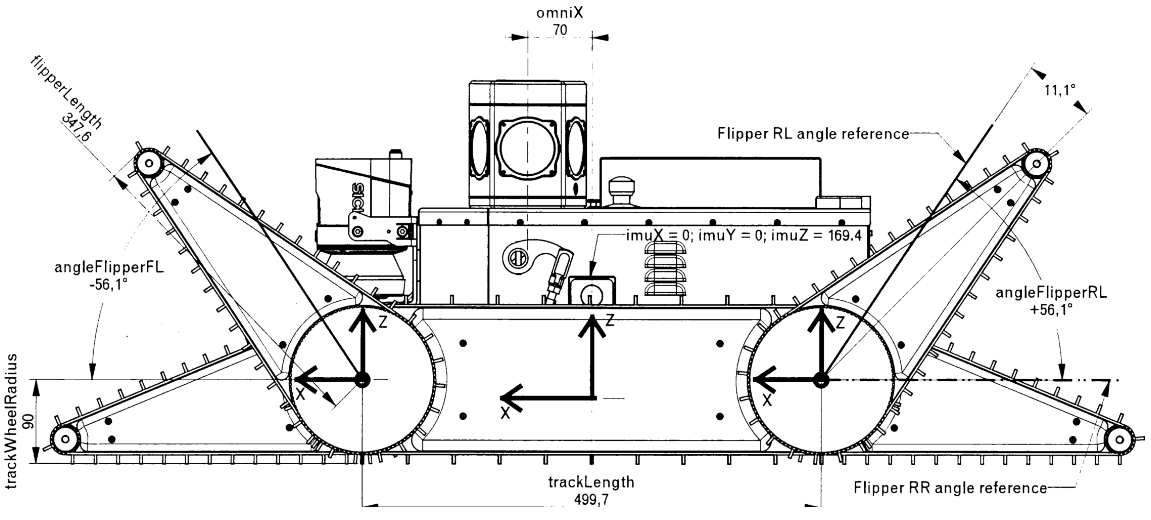


Figure 2.2: Side view of the robot

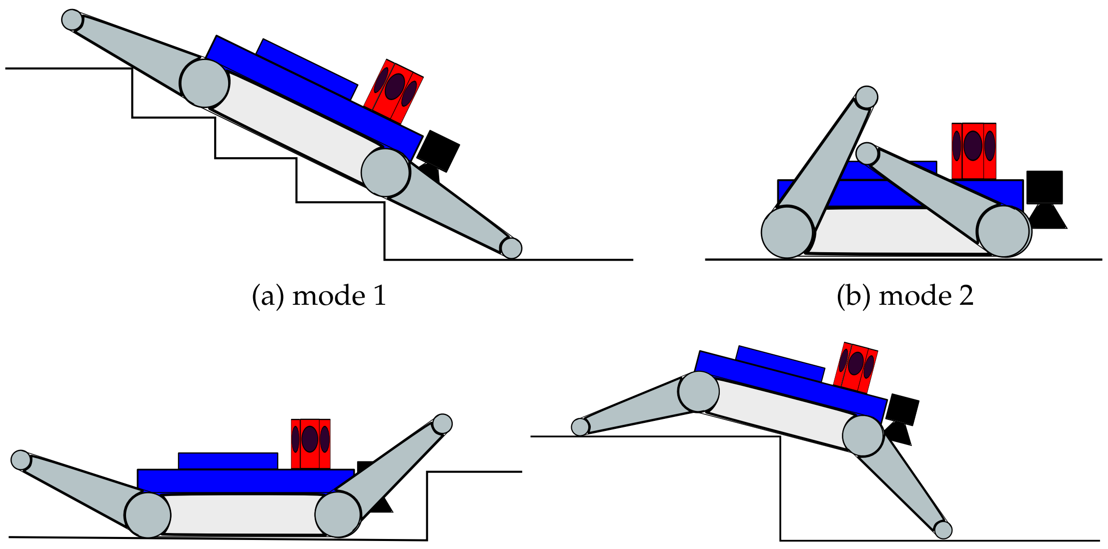
Inside the robot there is an embedded Kontron® PC equipped with the Intel® Core^{TM}2 Quad Mobile processor (Penryn) Q9100, which has sufficient computing power for a realtime image processing. Powered by a battery the robot can operate nearly 4 hours. For longer missions there is a possibility of a hot-swap battery replacement allowing an uninterrupted operation.

2.2 Robot Configuration Modes

The robotic platform we used has a great terrain traversability potential but it also has a large number of degrees of freedom which have to be controlled. To reduce the complexity of robot control, there are five predefined modes for the most common situations like for going up the obstacle, down the obstacle, etc. They are showed in the Fig. 2.3 and in detail described in table 2.1. Every mode is lateral symmetric which means that left and right flippers have the same angle.

Table 2.1: Front and hind flippers angles in different modes and its typical usage.

mode	typical usage	front [deg]	hind [deg]
1	unstable terrain with holes	0	0
2	compact for the best lidar view	-165	115
3	front obstacle	-45	10
4	back obstacle	-10	45
5	edge (peak) traversing	40	-40


(a) mode 1

(b) mode 2

(c) mode 3

(d) mode 5

Figure 2.3: Robot configuration modes. Mode 4 is reversed mode 3. Images taken from [1].

2.3 Software

In this section the third party software used in this work is described. It is mainly the open source software developed for the Robot Operating System (ROS) [15] by researches from all around the world including my colleges on CTU in Prague.

2.3.1 Robot Operating System

ROS is the state-of-the-art collection of tools, libraries and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms. It is maintained by the Open Source Robotics Foundation (OSRF) [16] which is an independent non-profit organization founded by members of the global robotics community. ROS provides an automatic management system for message-passing between nodes [17] using TCP/IP protocol. A node is an executable that uses the ROS to communicate with other nodes using topics [18]. Topics are named buses through which nodes exchange their data in a form of messages.

2.3.2 Octomap Server

Octomap server [19] is a ROS package based on the OctoMap library [20]. It builds and distributes volumetric 3D occupancy maps in various ROS-compatible formats. Created multi-resolution map represents occupied areas as well as free space. It is updatable with new sensor readings in a probabilistic fashion and dynamically expands if needed. Detailed information can be found in [21]. In this work it is used as an alternative representation of pointclouds obtained from a laser scanner. Points are recomputed to voxels from which centroids the DEM is computed. Example of an octomap representation is in the Fig 2.4.

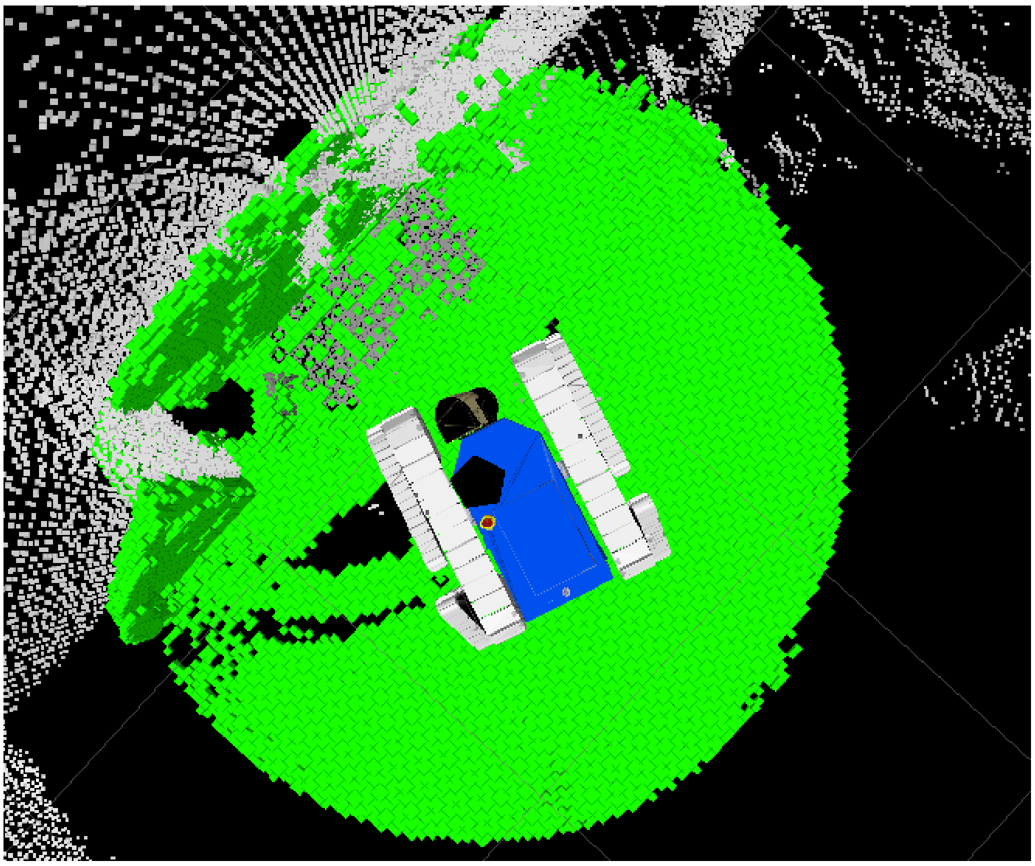


Figure 2.4: Pointcloud from laser data (grey) and octomap (green) build on its base after $360°$ robot rotation.

2.3.3 The Inertial Navigation System Aided by Odometry (INSO)

INSO [22] is a ROS node for the Extended Kalman filter based data fusion of odometry and inertial data (acceleration and angular rates) provided by IMU. This node provides a reliable dead reckoning navigation which is used in the octomap server for merging multiple 3D sensor measurements from different locations. In the future it will be extended by data from the visual odometry which will even improve the long-term accuracy.

2.3.4 GPML Matlab Toolbox

For a quick implementation of GP, the Matlab/Octave toolbox from [23] was used. It includes the main algorithms from [24] for regression and classification, containing set of mean, covarience and likelihood functions also as various inference methods. Everything from the previous is integrated into the function `gp()`, which does posterior inference, learns hyperparameters, computes the marginal likelihood and makes predictions.

3 Theory

In this chapter the theoretical analysis of methods used in this work is described.

3.1 Digital Elevation Map

Digital elevation map is an approximation of a real surface. A 3D sensor provides information about position of many points with a various spatial density. Because it is difficult to interpret these raw data, a suitable approximation is used. The surface is separated into grid cells of a same size. Elevation of each cell is the mean elevation of all points inside plus two times their variance. In this work, one cell is 10 x 10 cm which is a sufficient approximation of the terrain because the width of a main track of the robot used has also 10 cm. Complete DEM image used in this work (see the Fig. 3.1) consist of 20 x 5 cells, having the same width as the rover without flippers. It is defined in the robot body frame but with the compensation from the pitch and roll to stay parallel with the x-y plane of the global navigation frame, so it stays horizontal even if the robot inclines. It begins one meter before the front edge of the rover and continues one meter behind it.

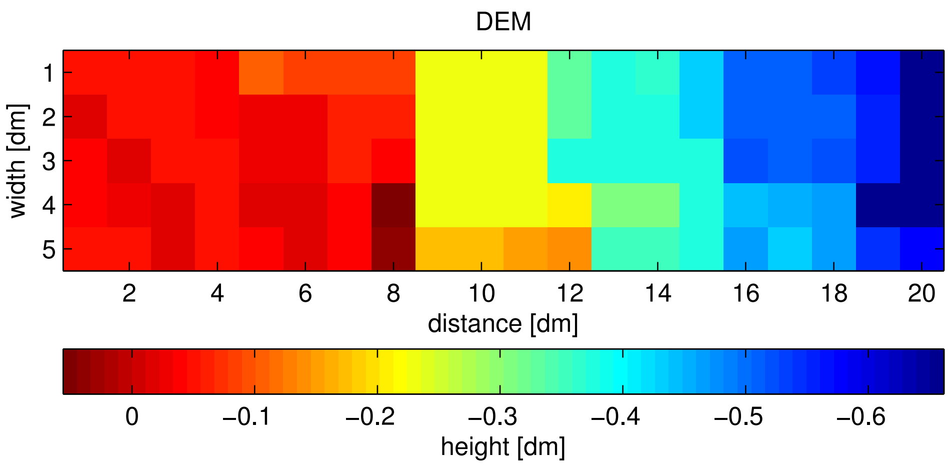


Figure 3.1: DEM image showing the stairs showed in the Fig. 3.2.

Because the rover footprint is a part of the DEM, it is invisible for the robot sensors. Actually, the sensor scans only a limited area in front of the rover. As the robot moves forwards, the known scanned area slides under the robot. The newly measured data are in the probabilistic manner joined together with those previously scanned using the octomap server (see the section 2.3.2) and fills the missing cells. The rest of the missing values is approximated from the neighborhood using a plane fitting. For better imagination the figure 3.3 shows the robot on the octomap of a same resolution as the DEM has (the octomap used for constructing the DEM had one order higher resolution).

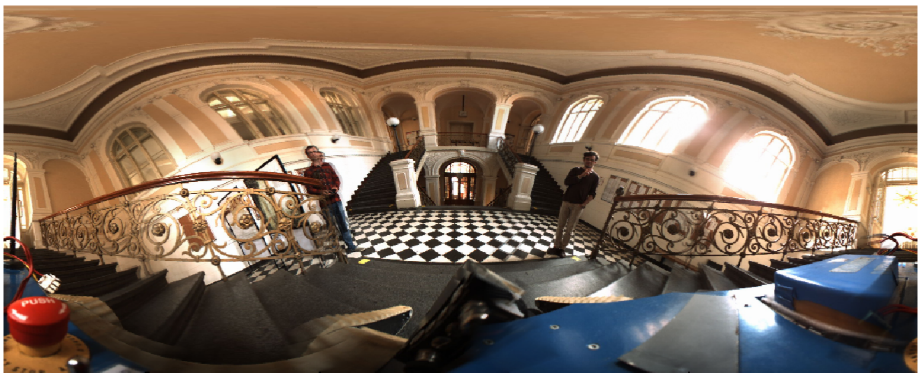


Figure 3.2: Stairs as seen from the robot omnicamera in a spherical projection.

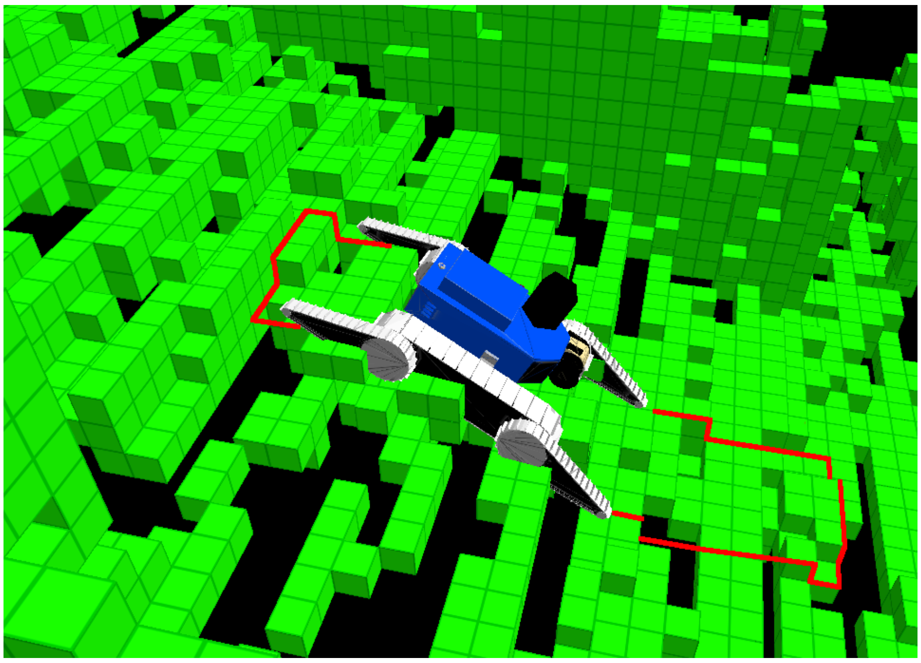


Figure 3.3: Robot on the similar stairs as in the Fig 3.1. Green voxels are obtained from the octomap server, red line is a border of the constructed DEM.

3.2 Features Extraction

DEM is processed in a similar way as the images in the machine learning. Instead of using pixels intensity values (heights of the DEM cells), computed Haar-like features are taken into account. The reason is that they are more general and robust to the noise. Finally, they reduce the dimension of the problem which leads to a faster algorithm operation. Computed feature vector contains Haar-like features computed from elevation of each DEM cell also as from the variance of detections inside each cell.

3.2.1 Haar-like Features

Haar-like features were introduced by Viola and Jones [25] as a fast feature for cascade face detector. In this work, only two kinds of edge features are used as shown in the Fig 3.4. The value of this feature is the difference between sums of heights of cells within two rectangular regions. These regions have the same size and shape and are horizontally (for pitch) or vertically (for roll) adjacent.

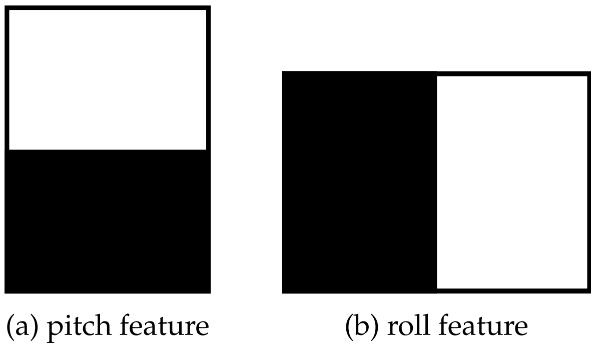


Figure 3.4: Haar edge features used to describe DEM

The fast way of computing Haar-like features uses computer vision method called integral image. The integral image at location x, y contains the sum of the heights of cells above and to the left of x, y, inclusive:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.1}$$

where $ii(x, y)$ is the integral image and $i(x, y)$ is the original image (DEM). Once the integral image is computed, it requires only three additions to compute the sum of heights of any rectangular area (see figure 3.5).

3.2.2 Feature Selection

Features are computed on the part of the DEM - called the region of interest (ROI) - which is affected by the robot footprint (see the figure 3.6). Because there is no prior knowledge of which features at which regions are suitable for the prediction, the features are computed over the all ROI and the greedy feature selection is used to determine, which features have a significant impact. It iteratively searches the full feature pool $\hat{\boldsymbol{X}}$ for the feature $\hat{\boldsymbol{X}}_{k^*}$ with lowest residual error $\Delta \boldsymbol{y}$ (`predicted value − reference value`) and adds it to the feature pool $\boldsymbol{X}$, which is then used in the next iteration. This method is inherited from [1] and is summarized in Alg. 1. The lower index $\boldsymbol{X}_k$ means the k-th feature from n features, the upper index $\boldsymbol{X}^i$ is the i-th sample from m samples.

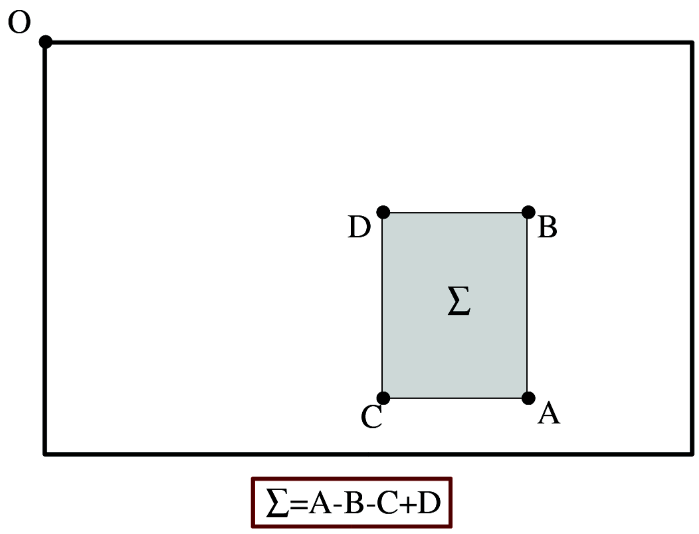


Figure 3.5: Using integral images, it takes only three additions to calculate the sum of heights inside a rectangular region of any size. The image is taken from [26].

3.3 Regression

Regression is one of the crucial method from the machine learning which is used for predicting continuous quantities. For given training samples (x, y) it tries to estimate the function $h(x)$ (hypothesis), that maps the input variable x to the target variable y and fits the training data with a minimal error. The function values of this function are called predictions. As there exist many functions there also exist many regression types reckoning desired function types.

In this work, the multivariate regression is used, which means that every output value y (pitch or roll) is computed from the input vector $\boldsymbol{x}$ called the feature vector. This vector consist of scalar quantities (features) computed from the DEM

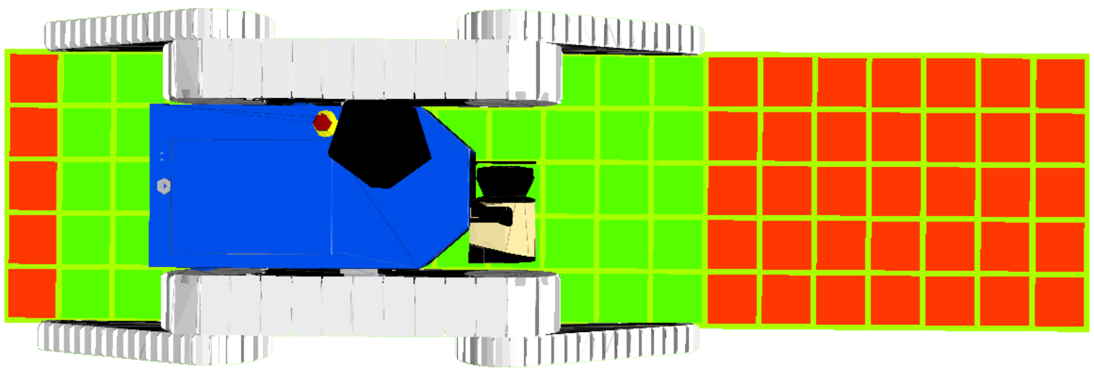


Figure 3.6: Robot on the ROI (green squares) of DEM (red squares) in the default position. ROI is used for computing Haar-like features. For predicting, the ROI is shifted forward by the length of prediction.

Algorithm 1 Feature selection procedure

```
   //Initialization
1: $\Delta \boldsymbol{y} = \boldsymbol{y}, \quad n = 1$
2: while (validation error is decreasing) do
     // Select the feature $\hat{\boldsymbol{X}}_{k^*}$ with the lowest residual error from the feature pool.
3:   $\hat{\boldsymbol{X}}_{k^*} = \arg\min_{(k)} \sum_{i=1}^{m} \|\boldsymbol{h}([\boldsymbol{X}_1^i \dots \boldsymbol{X}_{n-1}^i \hat{\boldsymbol{X}}_k^i]) - \Delta y^i\|^2$
     // Add the selected feature $\hat{\boldsymbol{X}}_{k^*}$ into $\boldsymbol{X}$.
4:   $\boldsymbol{X} = [\boldsymbol{X} \;\; \hat{\boldsymbol{X}}_{k^*}]$
     // Update residuals
5:   $\Delta \boldsymbol{y} = \boldsymbol{y} - \boldsymbol{h}(\boldsymbol{X})$
6:   $n = n + 1$
7: end while
```

(see the section 3.1).

3.3.1 Linear Regression

The basic regression model uses linear function. The hypothesis is defined by:

$$h(\boldsymbol{x}) = \Theta^T \boldsymbol{x} \tag{3.2}$$

where Θ^T is the transposed $(n+1) \times 1$ parameter vector, n is the number of features and x_0 from a $(n+1) \times 1$ feature vector $\boldsymbol{x}$ is equal to 1. The parameter vector is obtained in the learning phase from the training pairs in the form $(\boldsymbol{x}, y)$ using the closed form solution of the least squares problem called the normal equation method:

$$\Theta = \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \boldsymbol{X}^T\boldsymbol{y} \tag{3.3}$$

$$\Theta^T = \boldsymbol{y}^T\boldsymbol{X} \left(\boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \tag{3.4}$$

where $\boldsymbol{y}$ is a $m \times 1$ vector of target variables, m is the number of the training samples and $\boldsymbol{X}$ is the $m \times (n+1)$ feature matrix where each row corresponds to one feature vector from $(\boldsymbol{x}, y)$ pairs.

In cases where the redundant features (linearly dependent) are selected or when there is more features than training samples the $\boldsymbol{X}^T\boldsymbol{X}$ becomes non-invertible. Even pseudoinverse method based on the singular value decomposition (SVD) can fail. To solve this problem, the regularization [27] is used as follows:

$$\Theta^T = \boldsymbol{y}^T\boldsymbol{X} \left(\frac{\varepsilon}{\operatorname{cov}(\boldsymbol{y})}\boldsymbol{I} + \boldsymbol{X}^T\boldsymbol{X}\right)^{-1} \tag{3.5}$$

where ε is small number and $\boldsymbol{I}$ is the $(n+1) \times (n+1)$ identity matrix.

3.3.2 Piecewise Constant Regression

To approximate nonlinear functions, the piecewise constant (PWC) regression is used. PWC function $f(x)$ consists of B locally constant regions $f_i(x)$ joined together at breakpoints $b_1, b_2, \dots b_B$. In this work, these regions are equally sized except the border parts which are $(-\infty, \texttt{min\_value})$ and $(\texttt{max\_value}, \infty)$. Let the indicator function $C_E = 1$ when the event E is true and $C_E = 0$ otherwise. Then the regression model for $f(x)$ is:

$$f(x) = f_1 C_{\{b_0 \le x < b_1\}} + f_2 C_{\{b_1 \le x < b_2\}} + \cdots + f_B C_{\{b_{B-1} \le x < b_B\}} \tag{3.6}$$

Every feature from a feature vector $\boldsymbol{x}$ has its own PWC function which are aggregated together the same way as function parts in (3.6). The indicator function is then described using the $m \times B \cdot n$ indicator matrix $\boldsymbol{C}$ and the $m \times 1$ hypothesis vector $\boldsymbol{h}(\boldsymbol{X})$ can be obtained using:

$$\boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{C}\Theta \tag{3.7}$$

where Θ is $B \cdot n \times 1$ parameter vector. This parameter vector is obtained during the learning phase using the Moore-Penrose pseudo-inversion:

$$\Theta = (\boldsymbol{C}^T \boldsymbol{C})^{-1} \boldsymbol{C}^T \boldsymbol{y}. \tag{3.8}$$

Similarly to the linear regression (section 3.3.1), there is a problem with existence of inversion to the matrix $\boldsymbol{C}$. For that case, the regularization is made similar way:

$$\Theta = \left(\boldsymbol{C}^T \boldsymbol{C} + \frac{\varepsilon}{\operatorname{cov}(\boldsymbol{y})} \boldsymbol{I} \right)^{-1} \boldsymbol{C}^T \boldsymbol{y} \tag{3.9}$$

where ε is small number and $\boldsymbol{I}$ is the $n \times n$ identity matrix.

3.3.3 Gaussian Process Regression

GP regression is a general nonlinear interpolation method. GP defines a distribution over functions $p(f)$, where f is a function mapping from input space to output. From a definition, $p(f)$ is a GP if for any finite subset of inputs $\{x_1, \dots, x_m\}$ the marginal distribution over finite subset $p(\boldsymbol{f})$ has a multivariate Gaussian distribution. The $\boldsymbol{f} = (f(x_1), \dots, f(x_m))$ is an m-dimensional vector of function values evaluated at m points x_i. Every GP is parametrized by a mean function $\mu(x)$ and covariance function (kernel) $K(x_i, x_j)$:

$$p(f(x_1), f(x_2)) = \mathcal{N}(\boldsymbol{\mu}, \Sigma) \tag{3.10}$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} \mu(x_1) \\ \mu(x_2) \end{bmatrix} \tag{3.11}$$

$$\Sigma = \begin{bmatrix} K(x_1, x_1) & K(x_1, x_2) \\ K(x_2, x_1) & K(x_2, x_2) \end{bmatrix}. \tag{3.12}$$

In general, $\boldsymbol{\mu}$ is $m \times 1$ vector and Σ is an $m \times m$ matrix.

In this work, the kernel with squared exponential function with Automatic Relevance Detemination (ARD) distance measure is used. It is characterized by:

$$K(x^p, x^q) = \sigma_f^2 \mathrm{e}^{-\dfrac{(x^p - x^q)^T \boldsymbol{P}^{-1} (x^p - x^q)}{2}} \tag{3.13}$$

where the $\boldsymbol{P}$ matrix is diagonal with ARD parameters $l_1^2, ..., l_D^2$ (characteristic length-scales), where D is the dimension of the input space and σ_f^2 is the signal variance. The hyperparameters:

$$\theta = \{\log(l_1), \dots, \log(l_D), \log(\sigma_f)\} \tag{3.14}$$

can be learned from the marginal likelihood function (showed below). As the $l_i \to \infty$ the function f varies less and less on the i-th dimension and it becomes irrelevant. The ARD approach is used instead of feature selection (section 3.2.2) which is crucial for other types of regression. Thanks to this, the GP regression uses all features available.

For the regression assume the model:

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i \tag{3.15}$$

where $f \sim \mathcal{GP}(\cdot|0, K)$ and the noise $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. Considering the training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)_{i=1}^n\} = (\boldsymbol{X}, \boldsymbol{y})$ the predictions are then:

$$p(y_*|\boldsymbol{x}_*, \mathcal{D}) = \int p(y_*|\boldsymbol{x}_*, f, \mathcal{D}) p(f, \mathcal{D}) df = \mathcal{N}(\mu_*, \sigma_*^2) \tag{3.16}$$

$$\mu_* = \boldsymbol{K}_{\boldsymbol{y}*\boldsymbol{y}} \left(\boldsymbol{K}_{\boldsymbol{yy}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \tag{3.17}$$

$$\sigma_*^2 = \boldsymbol{K}_{\boldsymbol{y}*\boldsymbol{y}*} - \boldsymbol{K}_{\boldsymbol{y}*\boldsymbol{y}} \left(\boldsymbol{K}_{\boldsymbol{yy}} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{K}_{\boldsymbol{yy}*} + \sigma^2 \tag{3.18}$$

where the asterix $*$ marks the testing samples.

The marginal likelihood function can be also computed for comparing the covariance functions:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|f, \boldsymbol{X}) p(f) df = \mathcal{N}(0, \boldsymbol{K}_{\boldsymbol{yy}} + \sigma^2 \boldsymbol{I}) \tag{3.19}$$

$$p(\boldsymbol{y}|\boldsymbol{X}, \theta) = \mathcal{N}(0, \boldsymbol{K}_\theta + \sigma^2 \boldsymbol{I}). \tag{3.20}$$

The higher likelihood means better regression fit. Since it is a function of θ and σ, it can be optimized to find optimal hyperparameters θ. The closed form solution minimizes the log likelihood:

$$\ln\left(p(\boldsymbol{y}|\boldsymbol{X}, \theta)\right) = -\frac{1}{2} \ln\det(\boldsymbol{K}_\theta + \sigma^2 \boldsymbol{I}) - \frac{1}{2} \boldsymbol{y}^T (\boldsymbol{K}_\theta + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y} - \frac{\boldsymbol{\epsilon}}{2} \ln(2\pi) \tag{3.21}$$

More information about the Gaussian processes can be briefly found in the Zoubin Ghahramani's lecture [28] or detailed in [24].

3.4 RMS Error

To measure a quality of predictions, the Root Mean Square errors are computed and compared. It is defined by:

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n}(y_i - f(x_i))^2\right)} \tag{3.22}$$

where n is a number of samples, y is a desired value and $f(x)$ is a function value computed from a input value x.

4 Implementation

In this chapter implementation of algorithms proposed in this work is introduced, divided into these two main parts: learning and predicting. The data are recorded during the robot experiments, then processed in the MATLAB [29] where the regression coefficients are learned and then deployed for predicting the rover attitude.

4.1 Learning Phase

4.1.1 Extracting Data

One of the ROS (see the section 2.3.1) key features is the ability to record every topic (or their subset) on the robot and save them to a container called bagfile. It is done using ROS component rosbag [30] which can be used also for off-line replaying of these topics on any other system with ROS. During the experiment, the robot only records the sensory data to achieve a higher frequency of measurements. The DEM calculation (see the section 3.1), mapping and other commonly on-line running computations are made off-line on a desktop computer and the result is MATLAB readable file containing timestamped sequence of sensory data, the DEM and operators commands.

Building the Training Dataset

Each training sample consists of an angle acquired by the IMU and a vector of features obtained from a DEM ROI which is computed from 3D data acquired by lidar. It is important to note, that these two measurements (IMU and lidar) correspond to the same spatial area but are not acquired at the same time. Their precise alignment as a result of correct DEM composition in the octomap server using the navigation data from the INSO is therefore crucial for a quality of the machine learning. For the training samples the ROI lies under the robot footprint so the DEM features are calculated in the same time sample as the corresponding angle.

Building the Testing Dataset

The situation is more complicated for testing samples where the corresponding angle to the current DEM is in the future time sample. The solution is to bind data not by the time but by a distance traveled in the navigation frame which is computed during the data extraction for each sample by:

$$d_t = d_{t-1} + v_{\mathrm{x}_{t-1}} \cdot \Delta t \cdot \cos \theta_{t-1} \tag{4.1}$$

where d_i is the distance traveled from the beginning to the i-th sample in the navigation frame, v_x is the current robot speed according to the x-axis in the body frame obtained from the INSO node (see the section 2.3.3), Δt is a time difference between two samples and θ is the pitch angle.

Because the measured data were sampled almost equidistantly in the time domain, the dependency between sample index and time is almost linear. In contrast, the spatial domain is generally non-linear because the robot is not moving by a constant speed. As a next step, the testing samples are obtained as shown in the Alg 2. The input parameter is a prediction distance D which is an integer value specifying, in decimeters, how far the robot predicts (difference between the current and future position of the robot center of mass). This decimeters sampling was chosen because it is the size of one DEM cell and hence there is a minimal shifting value for the ROI over the DEM.

Algorithm 2 Testing sample composing procedure

```
Require: $0 \leq D \leq 7$ //the prediction distance in decimeters
 1: $D = \frac{D}{10}$ //convert it to meters to be in the same unit as the rest of distances
 2: idx $=1$
 3: for $i = 1$ to ( the number of input samples $m$) $- 1$ do
 4:   if $v_{\mathrm{x}_i} > 0$ and DEM ROI missing values cells $\leq T$ then
 5:     $\boldsymbol{F}$ = Haar-like features computed on the DEM ROI shifted by $10 \cdot D$ cells
        forwards in the direction of motion
 6:     for $j =$ i to $m$ do
 7:       if $d_j - d_i + D > -\epsilon$ and $d_j - d_i + D < \epsilon$ then
 8:         tst_data[idx].pitch $= \theta_i$
 9:         tst_data[idx].roll $= \psi_i$
10:         flipper_angles $=$ desired front and back flippers angles accord-
            ing to the mode at $i$-th sample
11:         tst_data[idx].pitch $=$ [$\boldsymbol{F}$, flipper_angles]
12:         idx $=$ idx $+1$
13:       end if
14:     end for
15:   end if
16: end for
17: return tst_data[]
```

The samples where the robot does not move are skipped because they do not provide any new information. Also the samples where the DEM ROI is incomplete (has more than empirically assessed $T = 2$ empty cells) are skipped because it is dangerous to drive the robot to a place of unknown elevation (sensor detected nothing). In a real application, these places would be marked as untraversable because it is probably a hole or a puddle, but it could be a very reflective or dark paving, safe for traversal as well.

The Haar-like features are computed on the DEM ROI (see the Fig. 3.6) which is shifted by the prediction distance (rounded to the DEM resolution) forwards in the direction of motion. This is the reason why the maximal prediction distance limit is 70 cm. The future corresponding sample is found by searching in the "traveled distance" fields d_i of future samples. This principle fails when the robot starts to move backwards, but this restriction was considered during the data collection, and in all experiments used in this work the robot moves only forwards. When searching for the corresponding samples a tolerance coefficient ϵ is taken into account because the sampling of the spatial domain is not continuous. The value of the ϵ was empirically set to be 5 cm which is a half of the DEM cell size. Once the corresponding pair is found, the features are associated with roll and pitch angles.

4.1.2 Model Training

Before training the linear and PWC model, the features are normalized to have zero mean and unit covariance. The reason is that during the training, the inversion of feature matrix is computed and the order differences causes a precision loss. As a next step, the feature selection is made, as described in the section 3.2.2.

The output of the learning phase consists of six trained models divided in two main groups, the first for a pitch angle and the other for a roll angle. Each group contains linear, PWC and GP regression model. The linear and PWC regression models contain a parameter describing selected features and parameters describing training features means and covariances, which are used for normalization the testing data. For computing the GP regression, training data must be in the used implementation provided as an input parameter, therefore these normalization parameters are computed online and are not stored in the model. The liner model has a parameter Θ, describing a magnitude of each feature, which is obtained using the equation (3.5).

The first PWC model parameter is B and describes a number of locally constant regions. This parameter was empirically set to 10, where the width of each region is 0.5 except for the border parts, which are on intervals $(-\infty, -2)$ and $(2, \infty)$. The second PWC parameter is Θ and describes a magnitude of each region and is obtained using the equation (3.9).

Thanks to the ARD, the GP regression does not need the feature normalization, nor the feature selection. However, it needs a hyperparameters vector which is obtained by a gradient descent optimization of the function (3.21). The GP model hence contains only these hyperparameters.

4.2 Predicting Phase

The trained models are used for predicting the angles. Before the regression is made, features are normalized according to a training dataset:

$$F_{norm_i} = \frac{F_i - \mu_i}{\sigma_i^2} \tag{4.2}$$

where F_i is the i-th testing feature, μ_i is a mean value of the i-th training features and σ_i^2 is a variance of the i-th training features. Than the linear regression is computed using the equation (3.2), the PWC regression using the equation (3.7) and the GP regression using the equation (3.16).

5 Evaluation

5.1 Datasets

To provide machine learning with training and testing data, the expert operator traversed the robot repeatedly across urban-type obstacles: wooden pallets and various types of stairs (see the Fig. 5.1). Most of experiments used in this work were made for the autonomous traversability learning [1], where the requirements for a dataset are similar. Each single experiment (called maneuver) is formed by a situation where the robot is successfully moving forwards across the obstacle using multiple modes (see the section 2.2). There are, in total, 17 maneuvers used for training, that consist of crossing the wooden pallets. Further there are two levels of testing: *simple* and *complex*. The *simple* testing consists of two maneuvers where the robot crosses pallets. The *complex* testing consists of a single long maneuver at stairs that were not used for training. This long maneuver is in the Fig. 5.2, where the stair hall from Fig. 3.2 is traversed. The roll and pitch angles from all datasets are in the Fig. 5.3, Fig. 5.4 and Fig. 5.5.

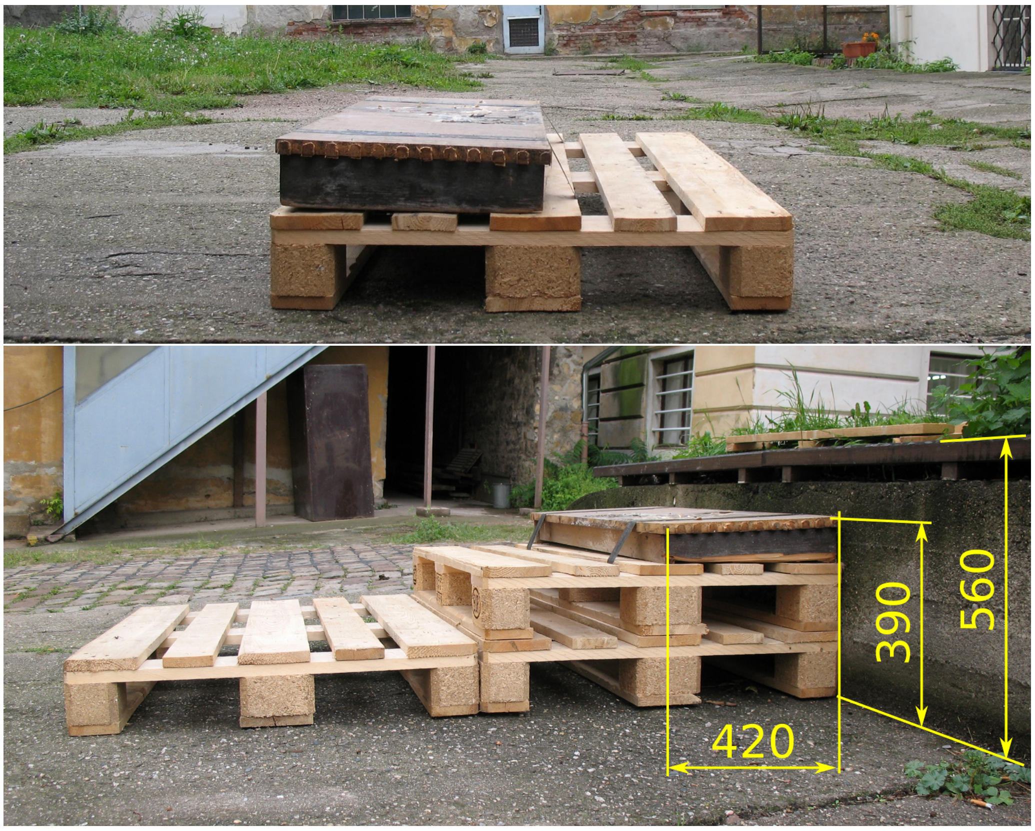


Figure 5.1: The wooden pallets used for training which images are taken from [1]

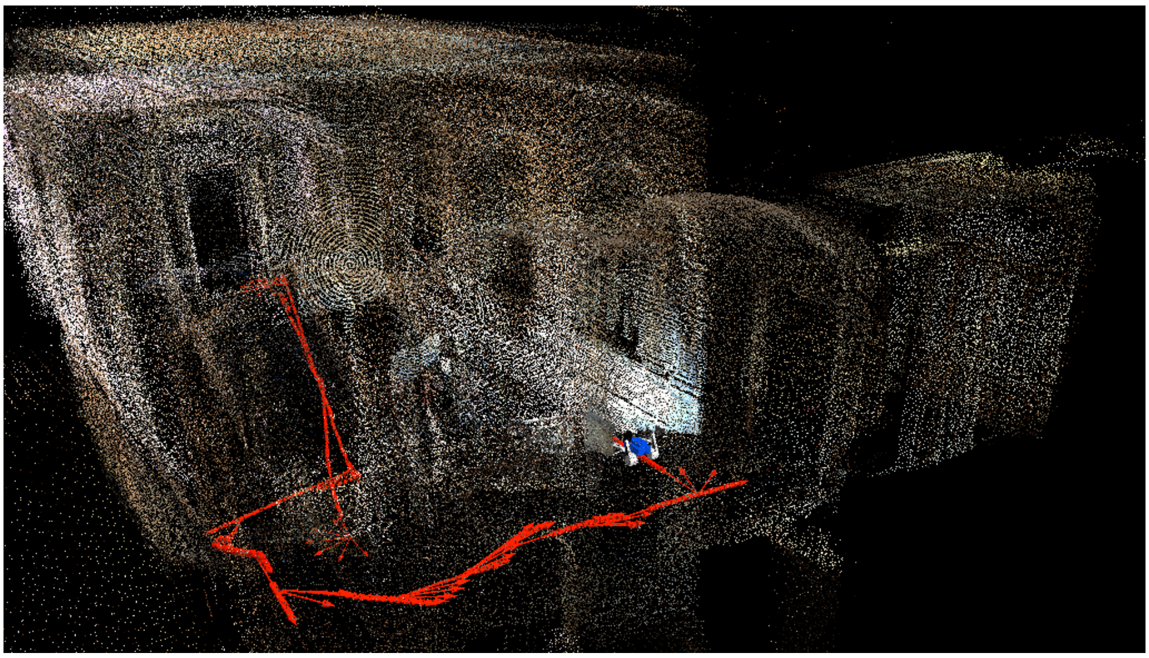


Figure 5.2: Robot trajectory (red arrows) during the long maneuver at stairs.

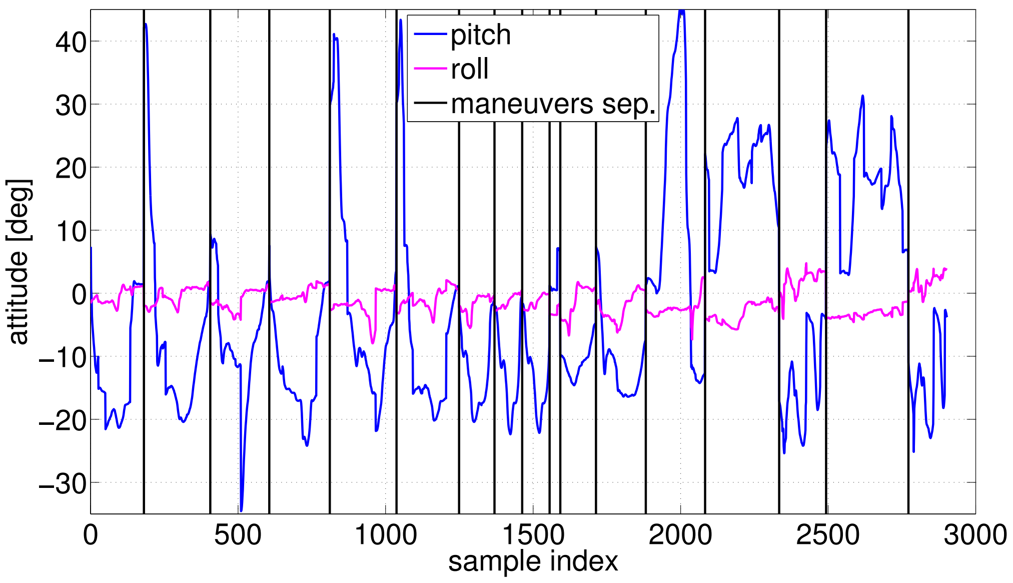


Figure 5.3: Training data

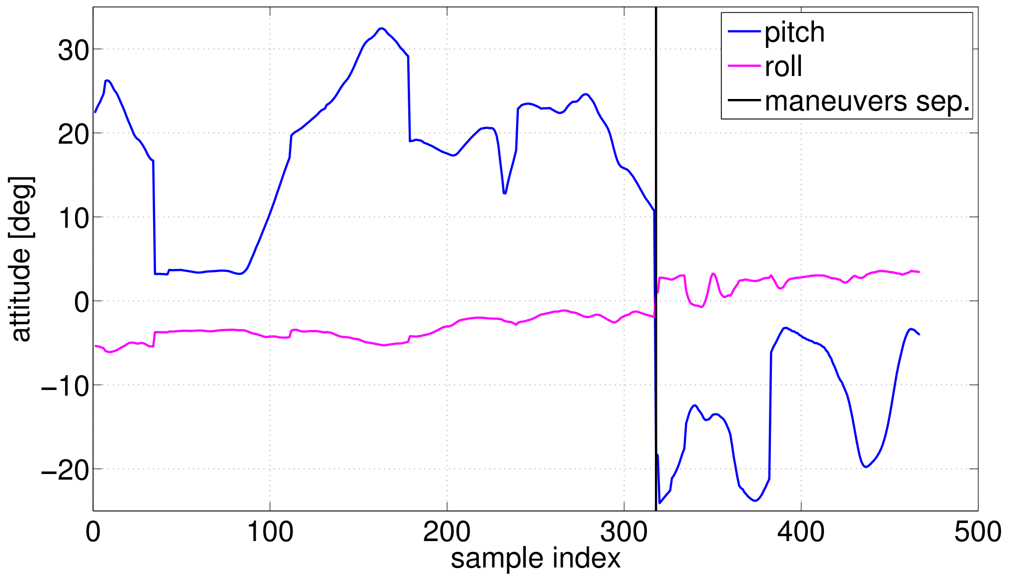


Figure 5.4: Simple testing data

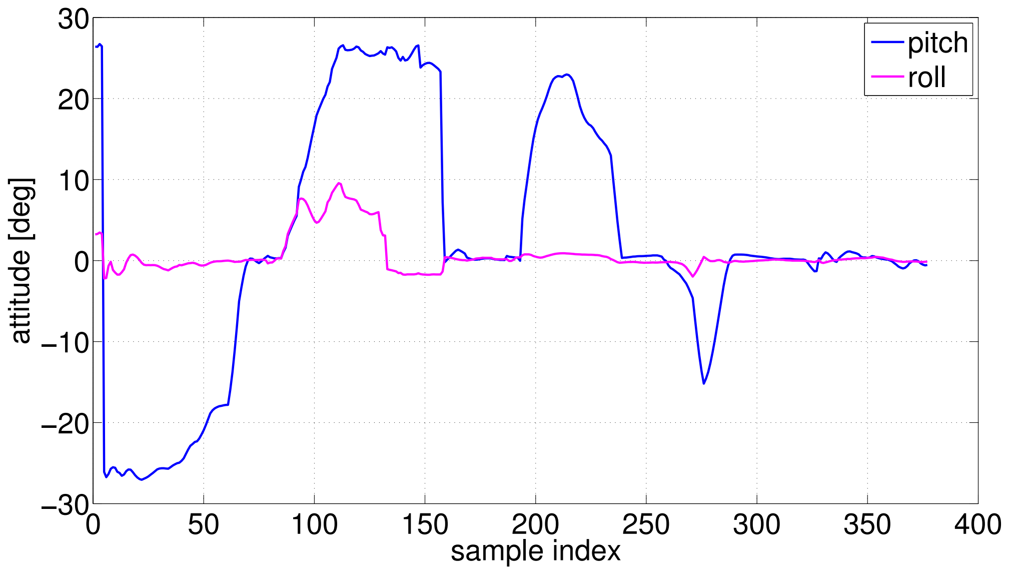
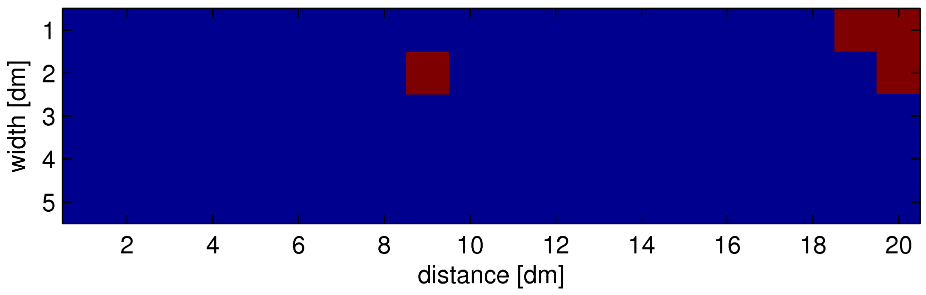


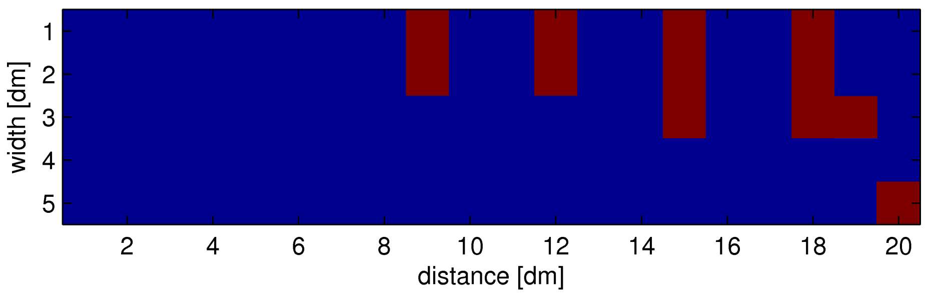
Figure 5.5: Complex testing data

5.1.1 Incomplete DEM

For predictions at longer distance there is a problem with incomplete DEM, as shown in the Fig. 5.6. Although plain fitting technique is used to cover the missing cells, the real surface is uncertain. Therefore, a restriction for maximal number of the missing cells was set that maximally two of them are allowed in the ROI. This leads to a diminishing size of datasets for longer prediction, as shown in the Fig. 5.7.



(a) Almost flat surface



(b) Stairs

Figure 5.6: Missing values in DEM (red cells)

5.2 Proof of the Concept

To verify the possibility of predicting the position angles according to the feature vector chosen in this work (which means that feature vector contains features, which are altogether sufficient for a correct computation of position angle) the prediction of the current measured attitude from the newest previously measured data is made. This verification is done on data obtained in the same way as the training data, but on testing dataset, i.e., angles measured by IMU are compared to angle computed from DEM ROI in the same time sample. It is not the time sample, when the data were acquired by a 3D sensor (they were acquired when the robot was a half its longitude backwards). The prediction is computed with a zero shift of the ROI against the DEM and therefore the Alg. 2 is not used. Because the DEM ROI contains the newest data that are available, it is expected, that the prediction error will be the lowest possible. Therefore a definite number of features suitable

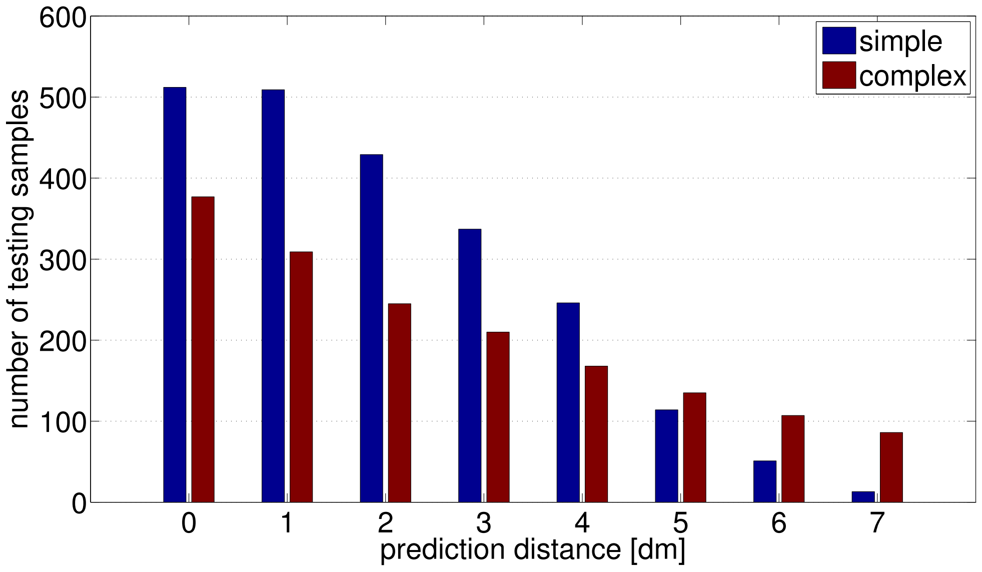
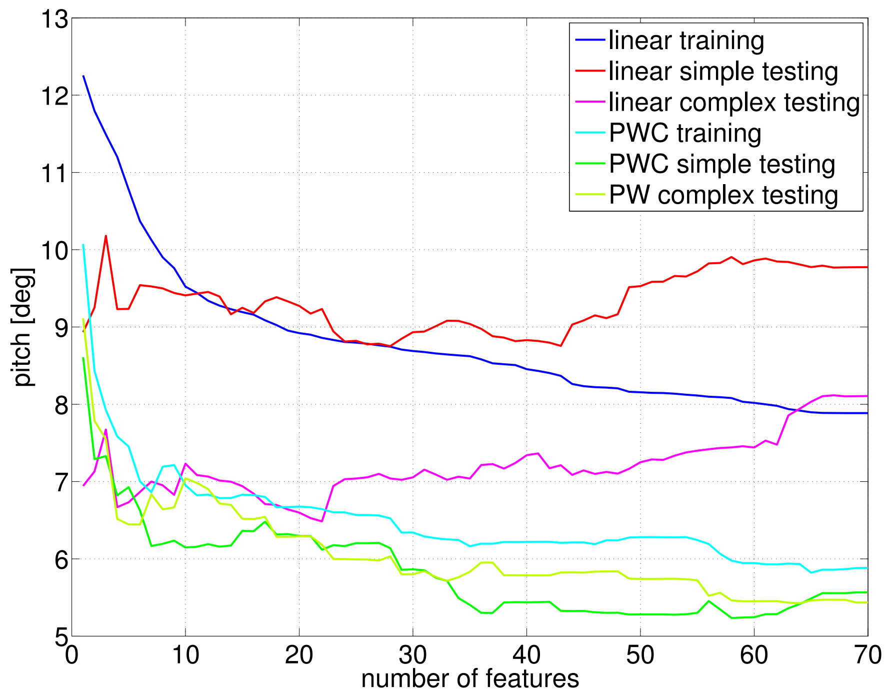


Figure 5.7: Diminishing number of training samples with longer prediction distance.

for linear and PWC regression, which minimizes testing RMS error, is chosen, and this number of features is used for other distances, where this prediction is used.

The Fig. 5.8 shows the RMS errors for linear and PWC regression for the pitch and roll angles. The optimal number of features based on this experiment is set as 28 for the linear regression (both angles), and 60 for the PWC regression of pitch and 15 for the PWC regression of roll. It should be noted that these numbers are not strict, because selecting for example 40 features for the PWC regression of pitch does, with respect to the angles measurement accuracy, almost the same prediction errors. The linear regression results are in the Fig. 5.9 and 5.10, PWC regression in the Fig. 5.11 and 5.12, and the GP regression in the Fig. 5.13 and 5.14. The GP regression also estimates variance of its prediction which is plotted in the figures as a 2σ neighborhood.



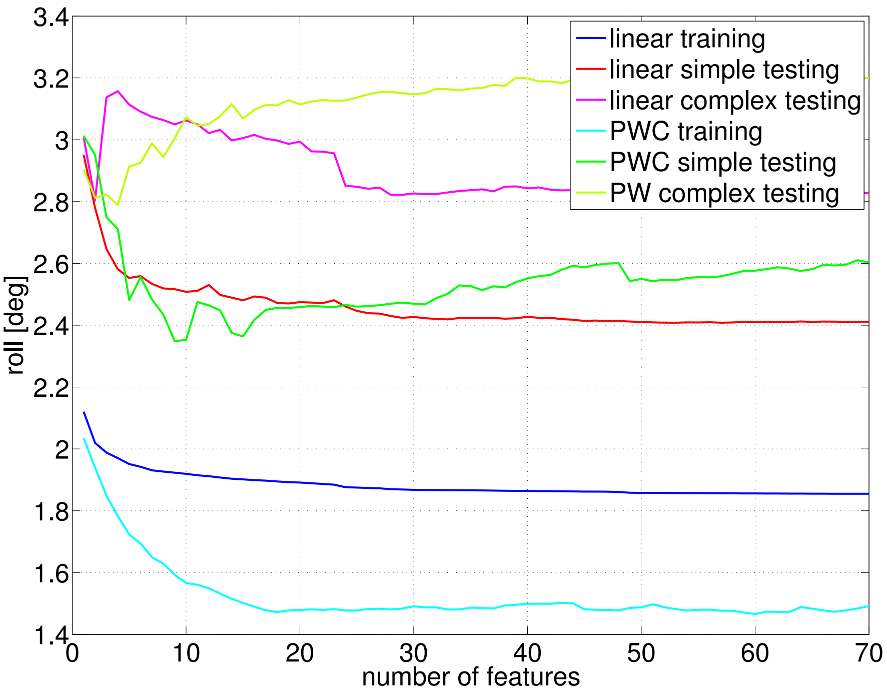
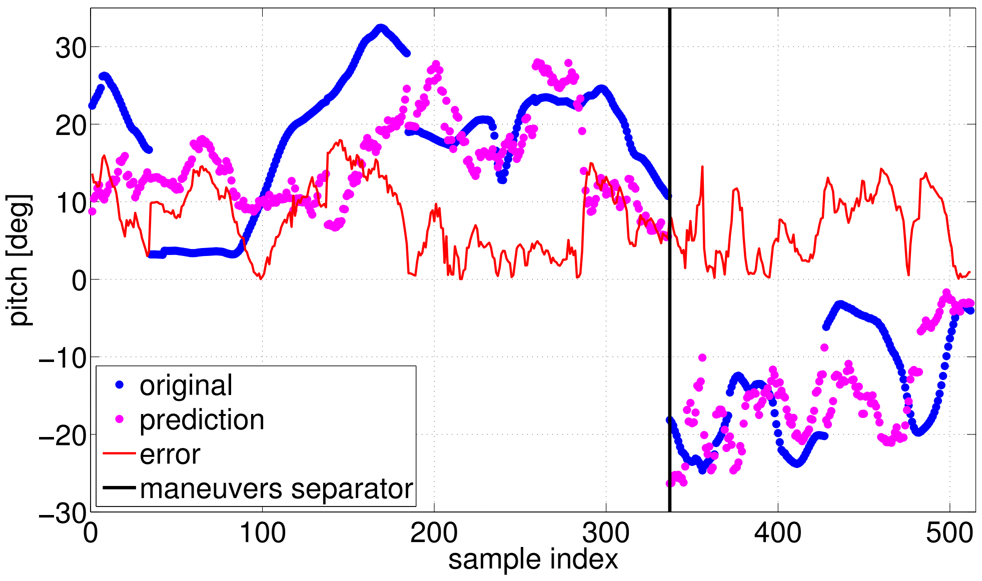


Figure 5.8: Linear and PWC regression RMS errors for various number of features. GP regression uses all features as mentioned in the section 3.3.3.



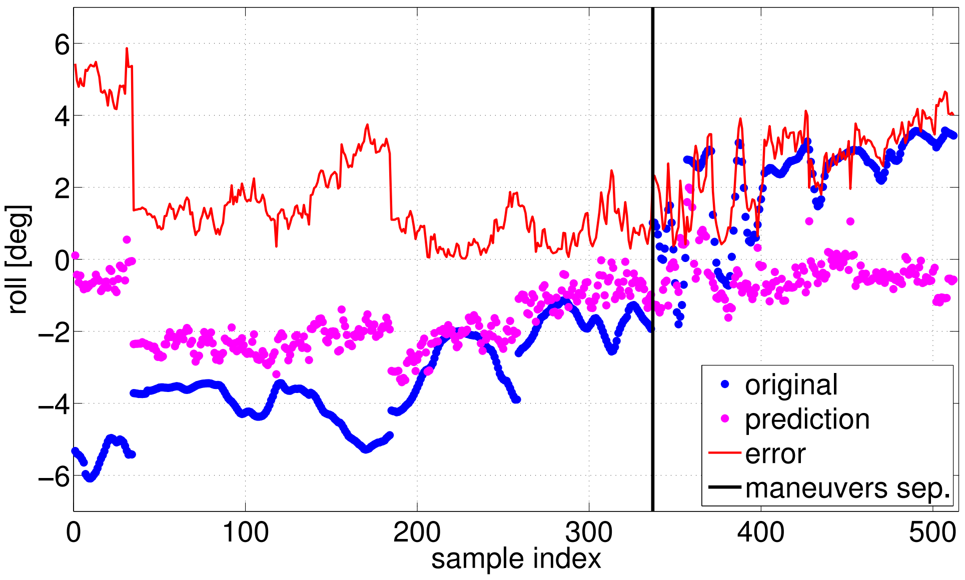


Figure 5.9: Linear regression for the zero ROI shift on the simple testing data.

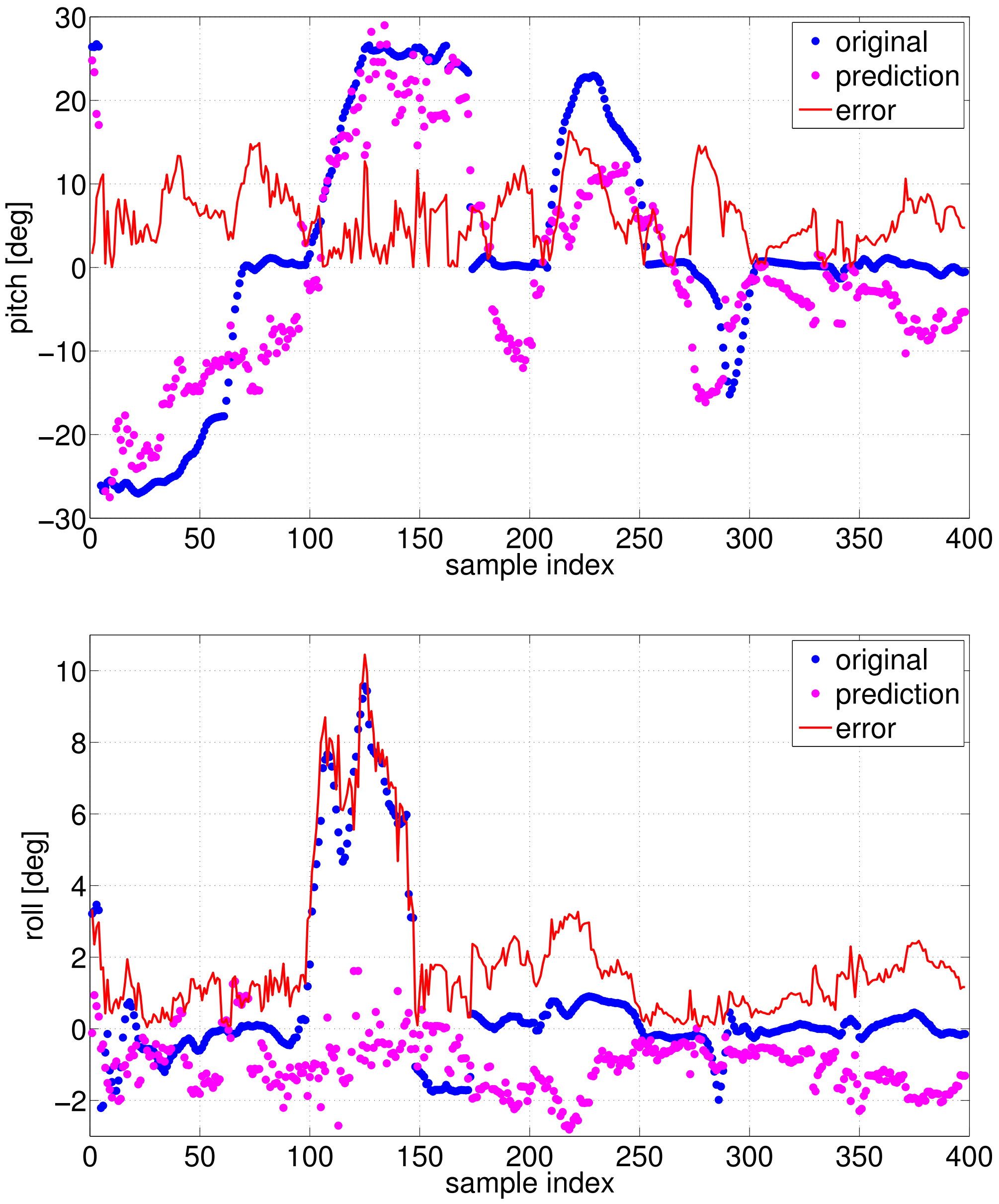


Figure 5.10: Linear regression for the zero ROI shift on the complex testing data.

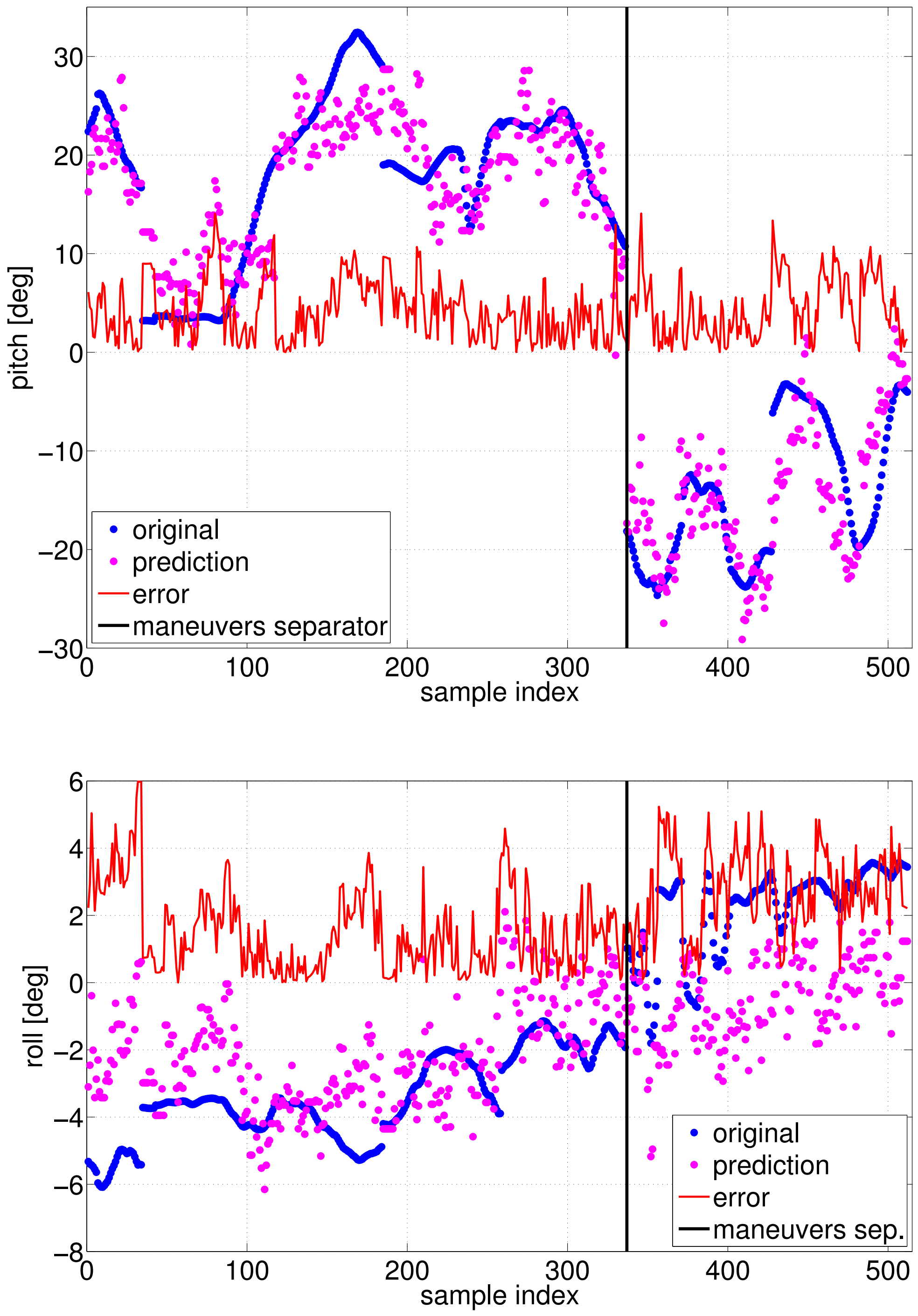


Figure 5.11: PW regression for the zero ROI shift on the simple testing data.

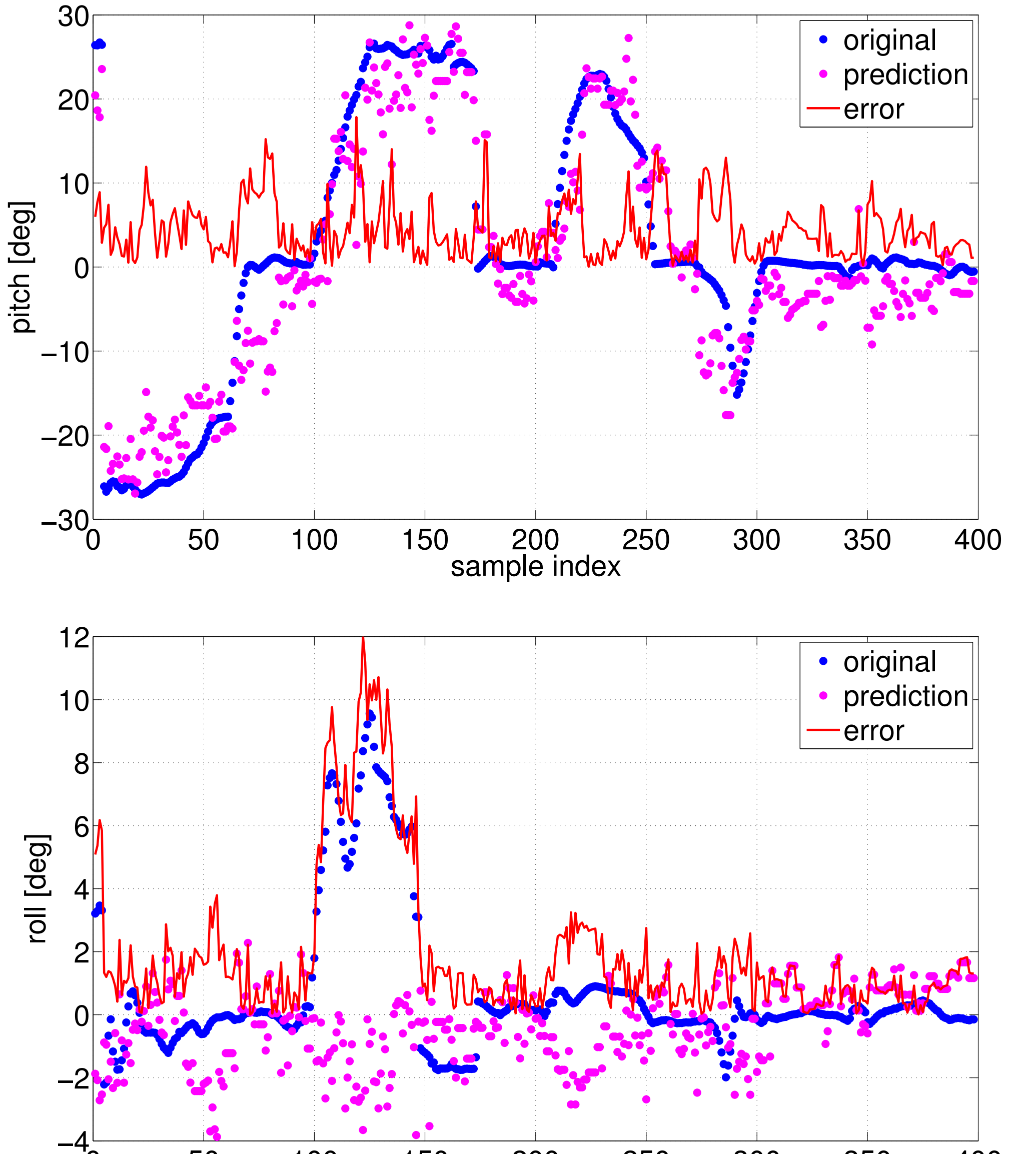


Figure 5.12: PW regression for the zero ROI shift on the complex testing data.

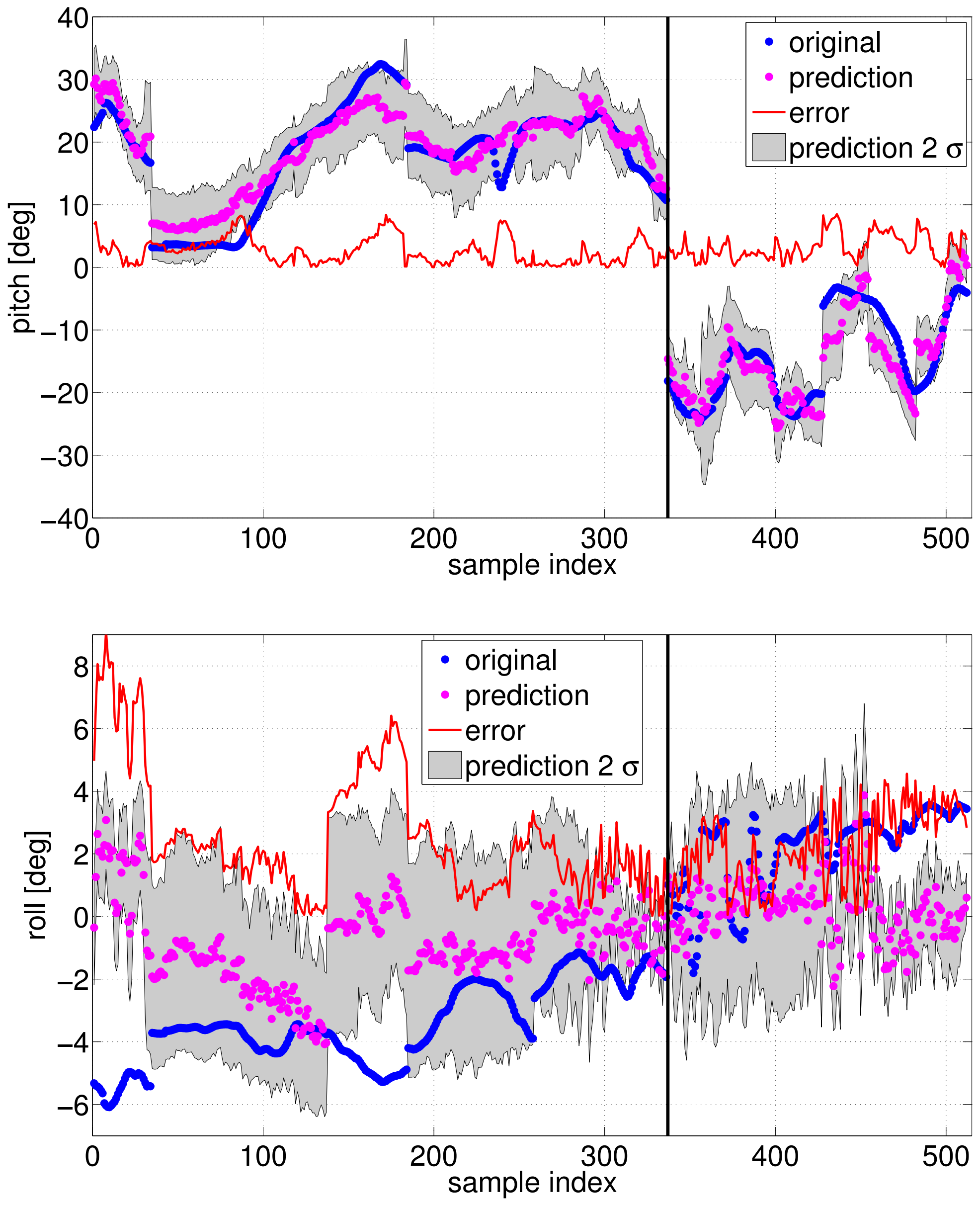
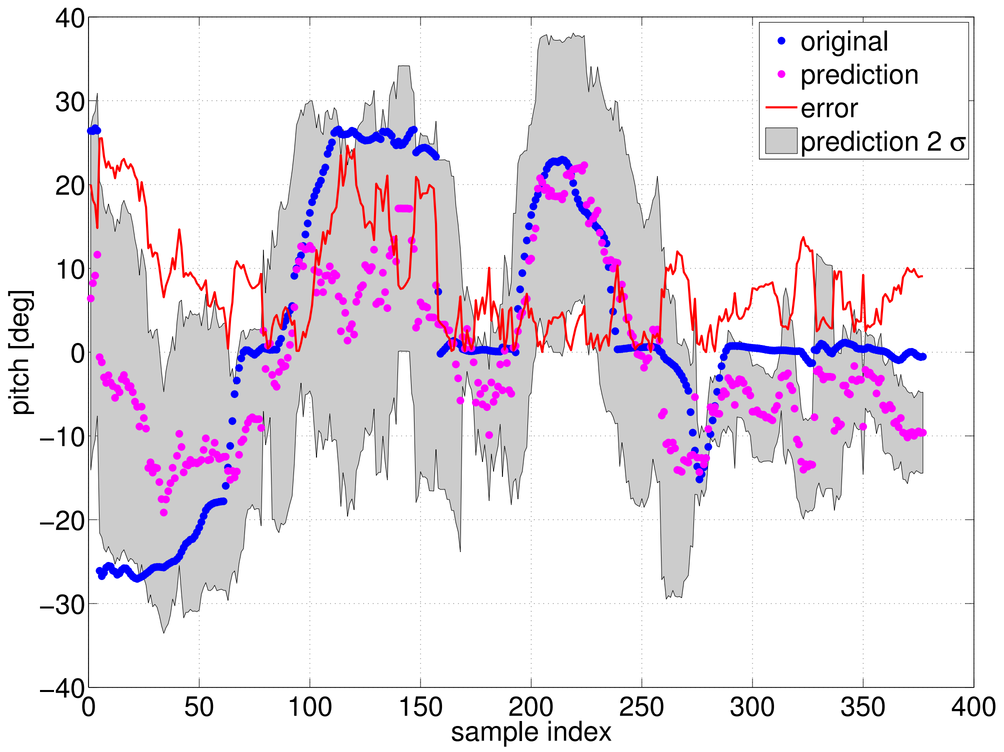


Figure 5.13: GP regression for the zero ROI shift on the simple testing data.



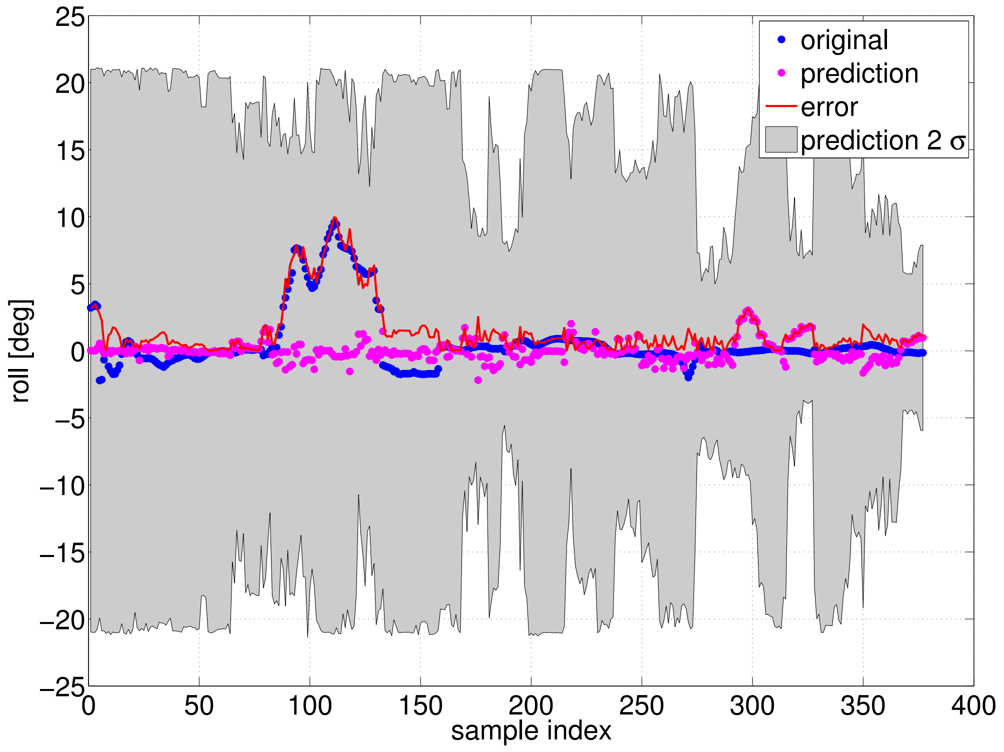


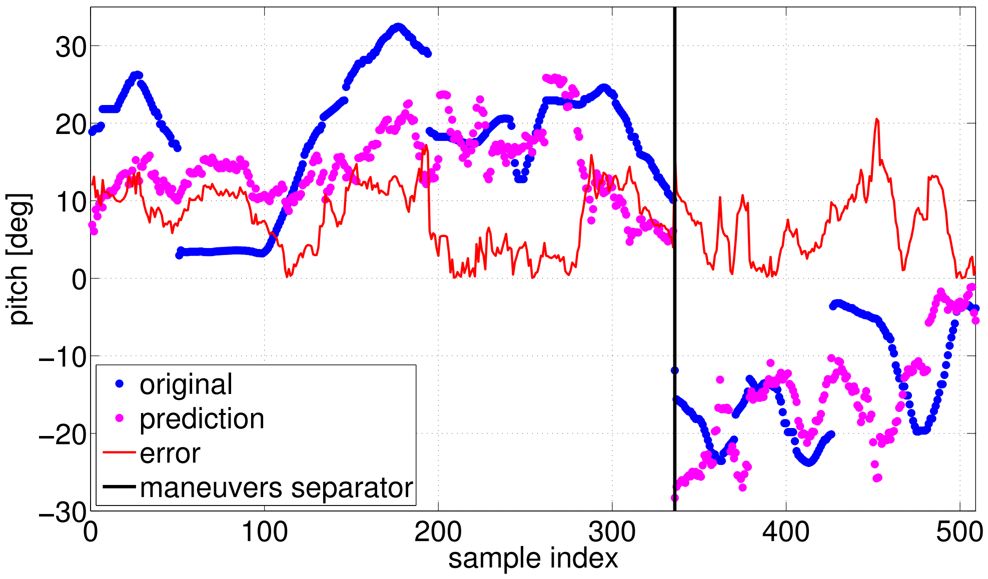
Figure 5.14: GP regression for the zero ROI shift on the complex testing data.

5.3 Results

The next sections describe the experimental results of all trained models on the testing (simple and complex) datasets for pitch and roll for individual prediction distances. Prediction distance is the difference between the actual and future position of the robot center of mass. The table 5.1 summaries all the results. The lower errors for the roll angle are a consequence of lower angle deviations in the training and testing dataset. It is also a reason for the better results of the linear regression in predicting roll angle, because non-linear property of the used features is insignificant.

5.3.1 Linear Regression

Fig. 5.15-5.22 show the linear regression predictions for distances from 10 cm to 40 cm for simple and complex testing dataset for pitch and roll.



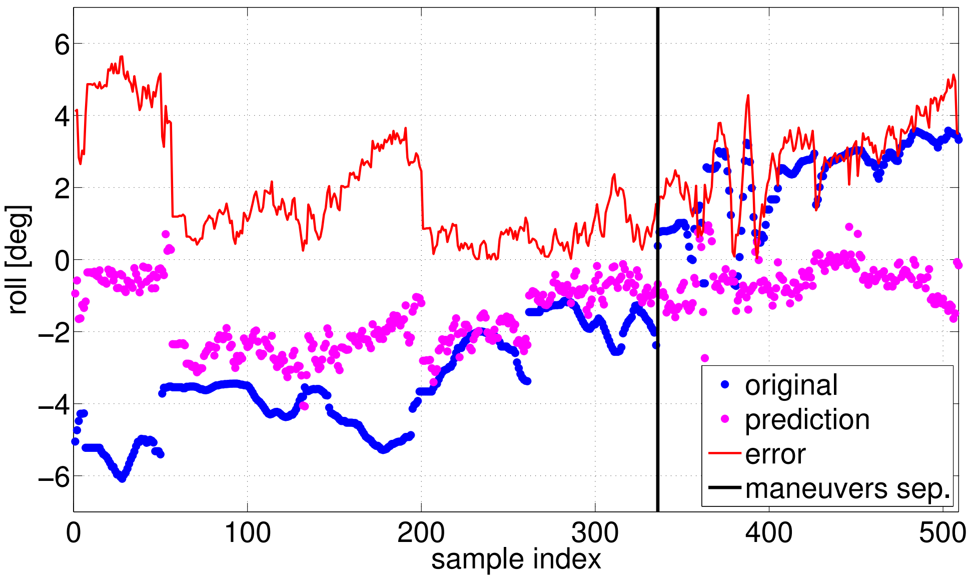


Figure 5.15: Linear regression for the 10 cm distance on the simple testing data.

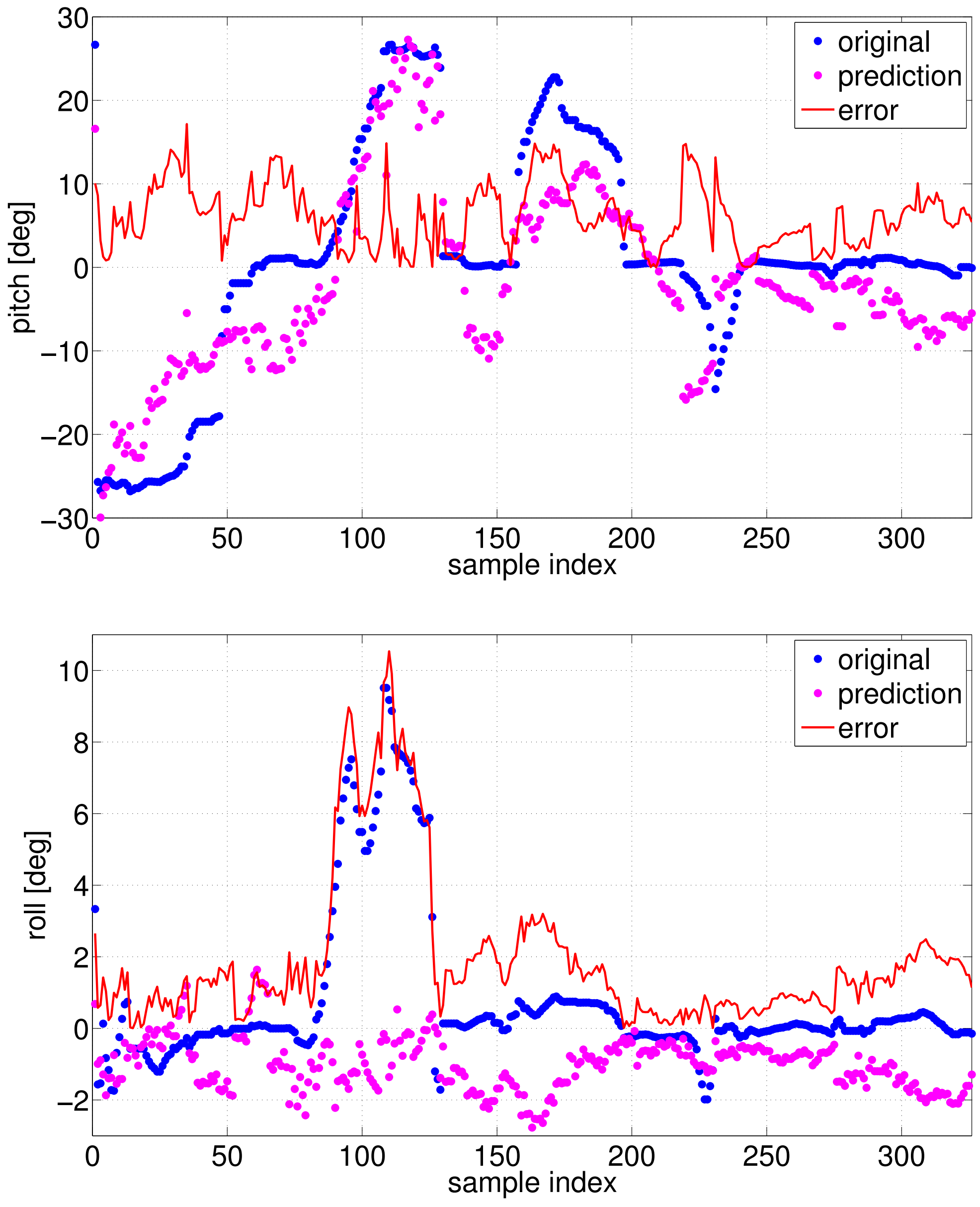


Figure 5.16: Linear regression for the 10 cm distance on the complex testing data.

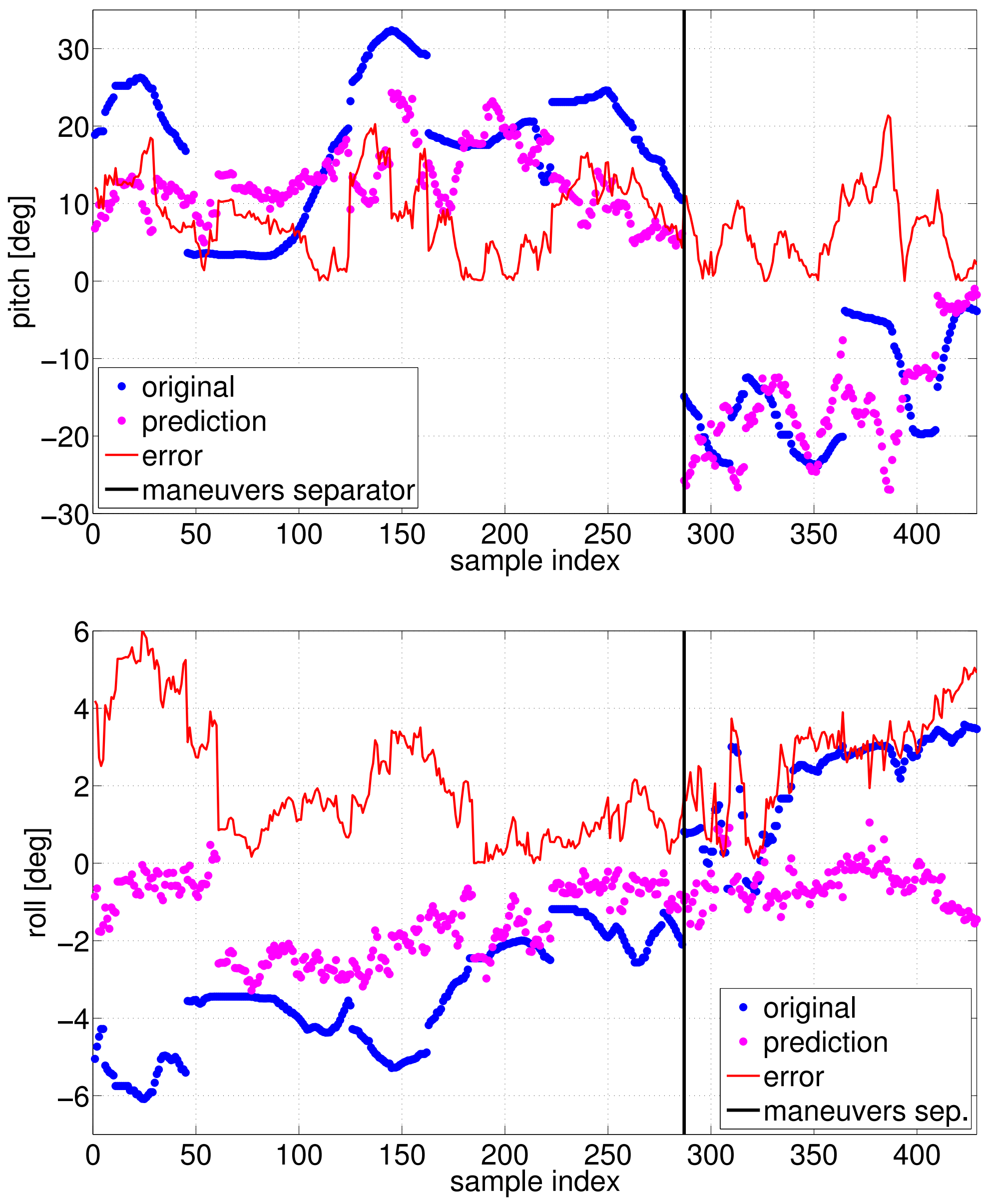


Figure 5.17: Linear regression for the 20 cm distance on the simple testing data.

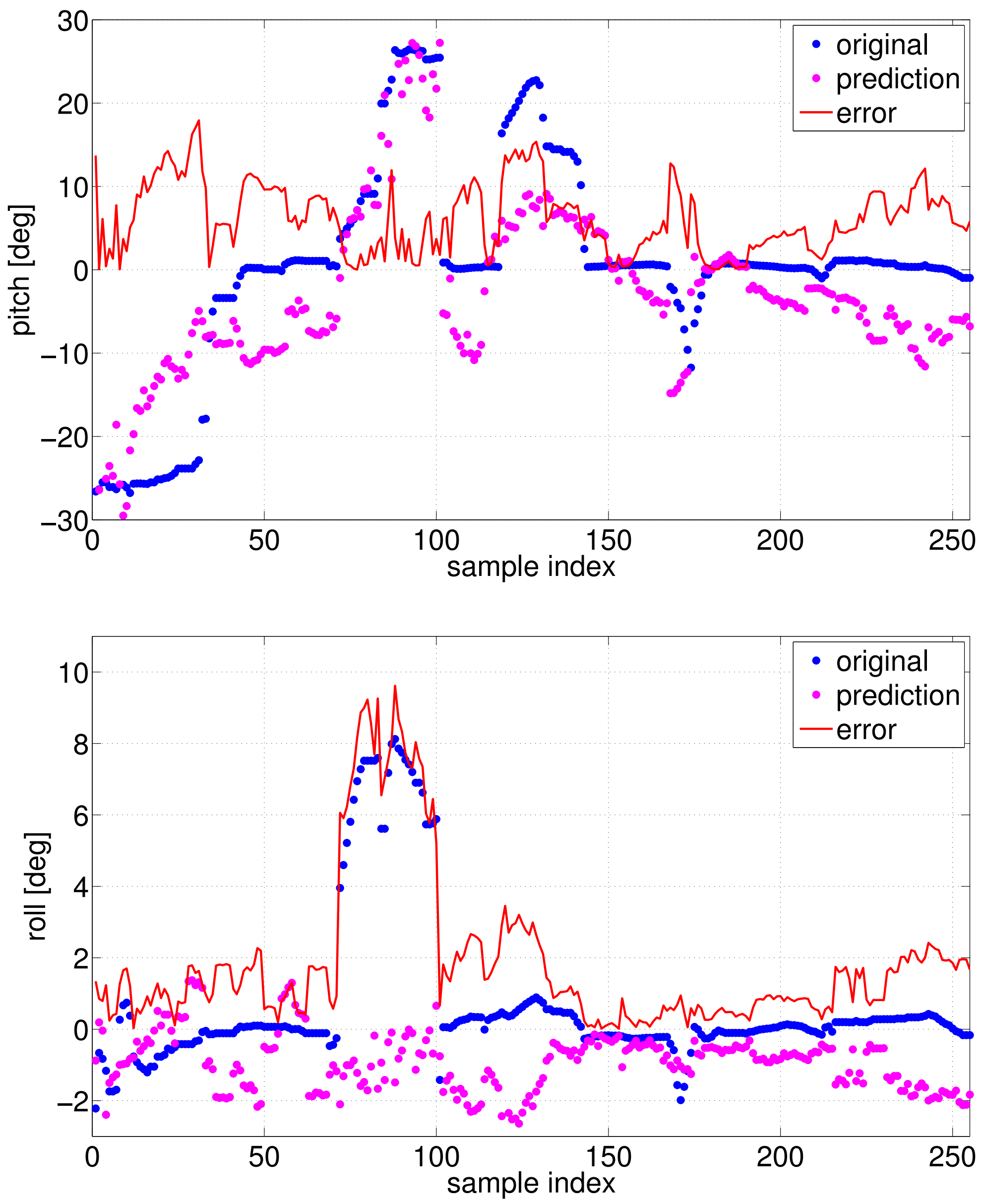


Figure 5.18: Linear regression for the 20 cm distance on the complex testing data.

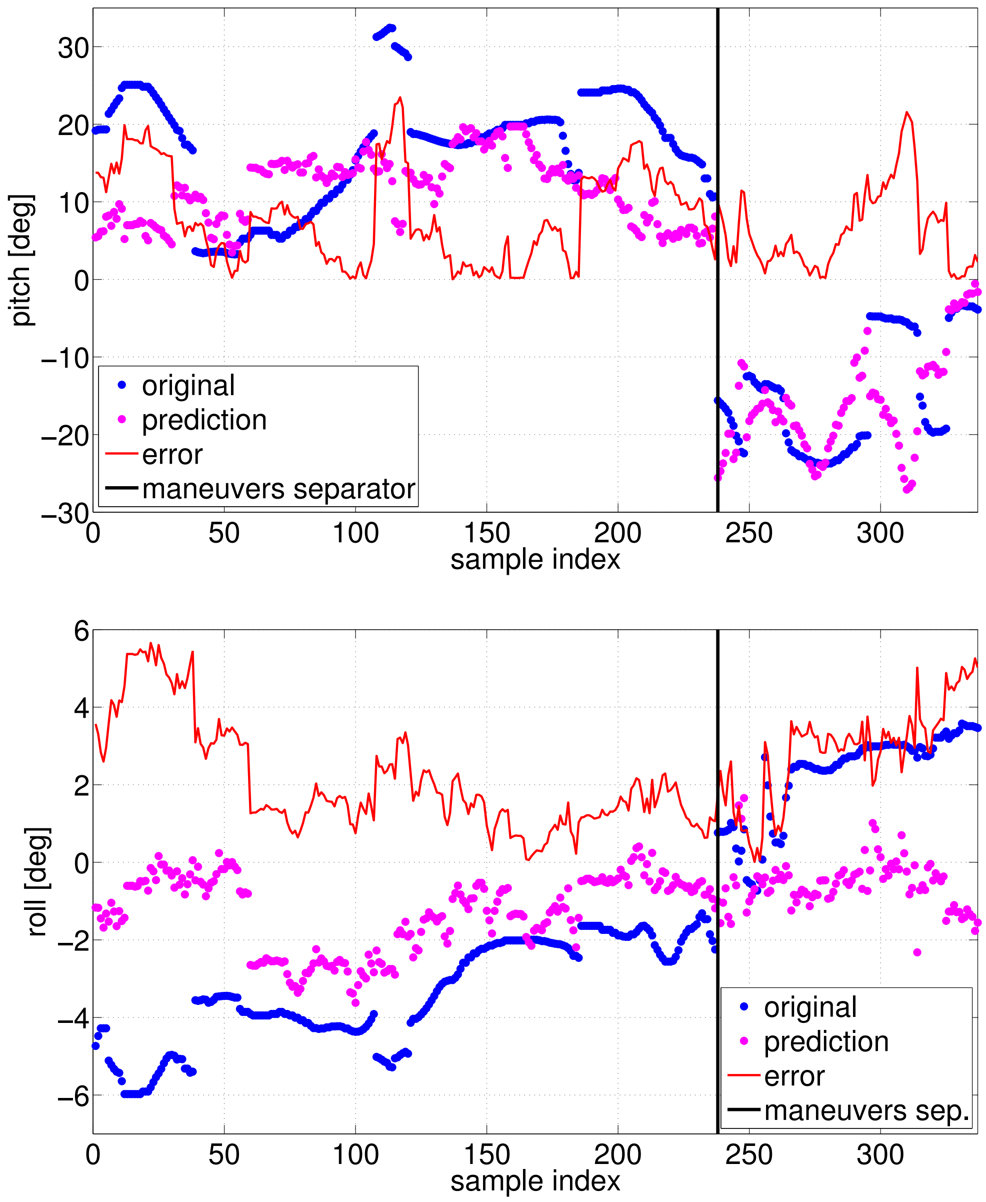
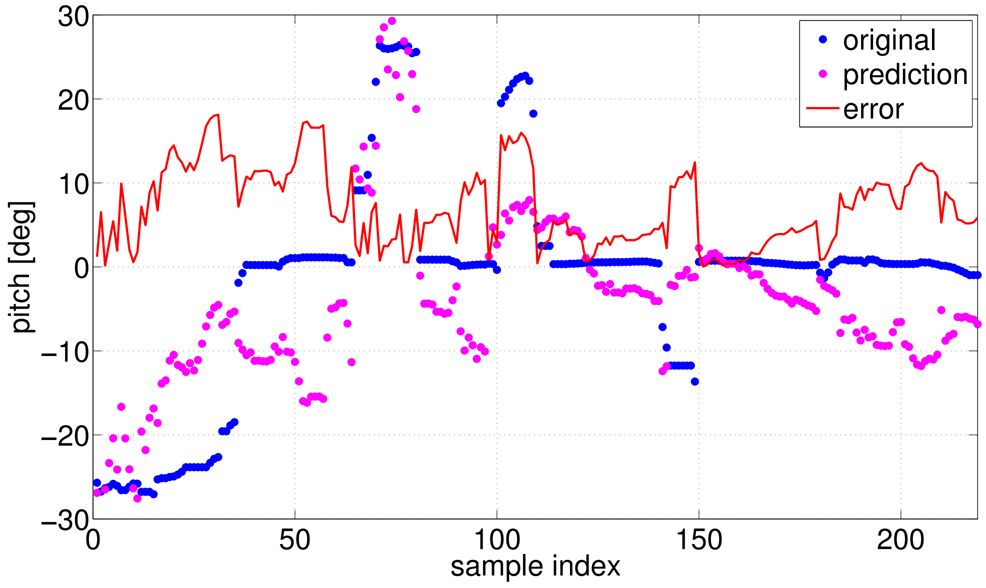


Figure 5.19: Linear regression for the 30 cm distance on the simple testing data.



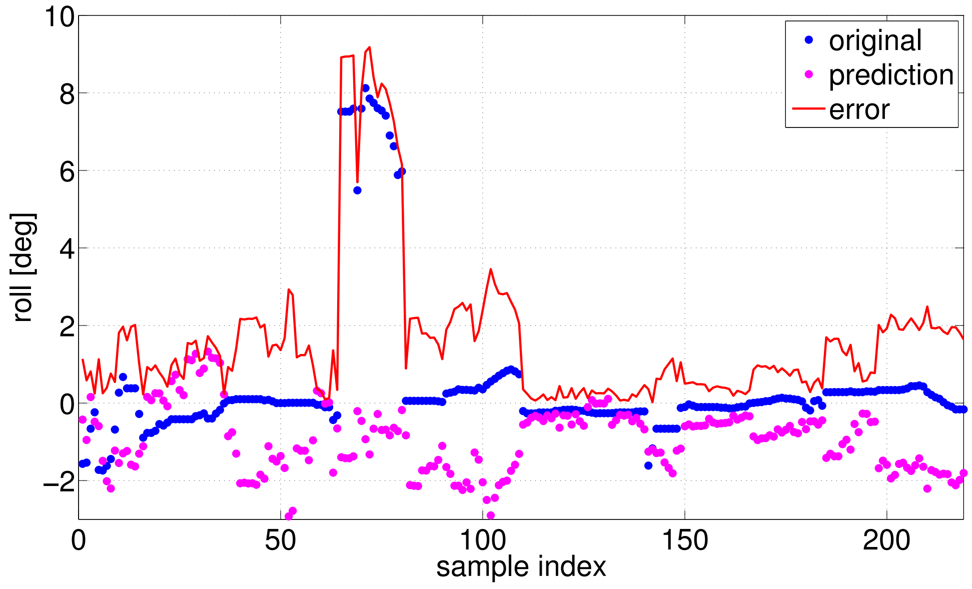
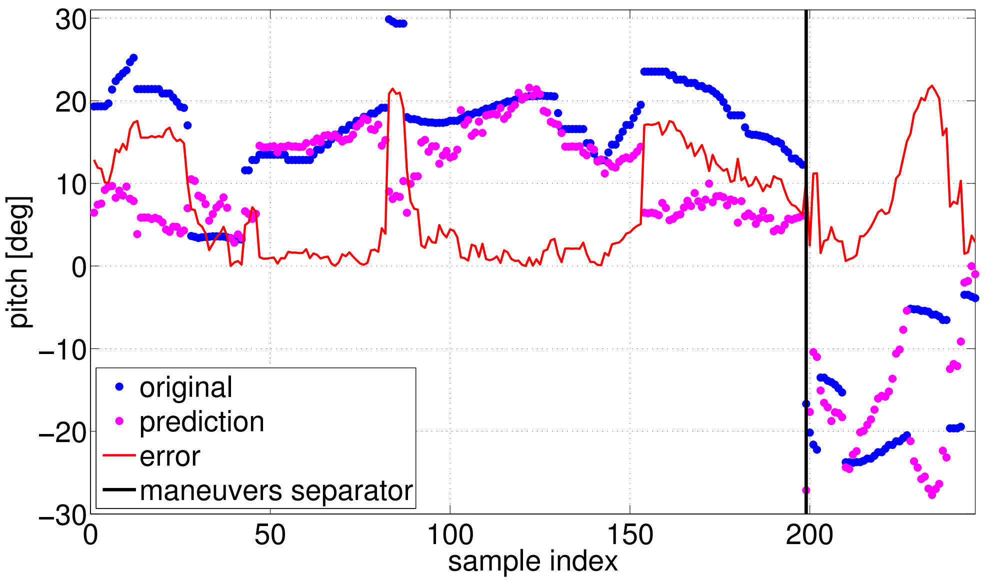


Figure 5.20: Linear regression for the 30 cm distance on the complex testing data.



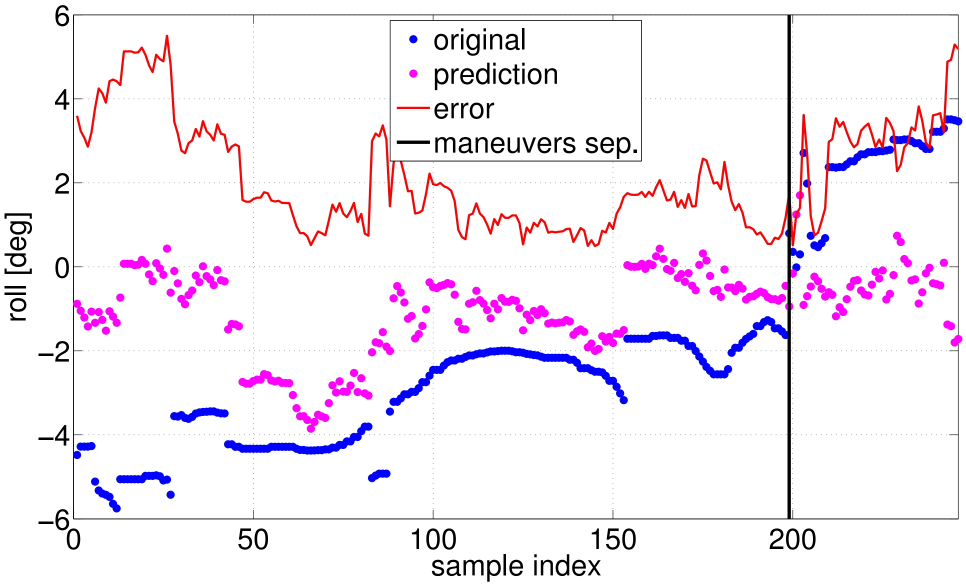


Figure 5.21: Linear regression for the 40 cm distance on the simple testing data.

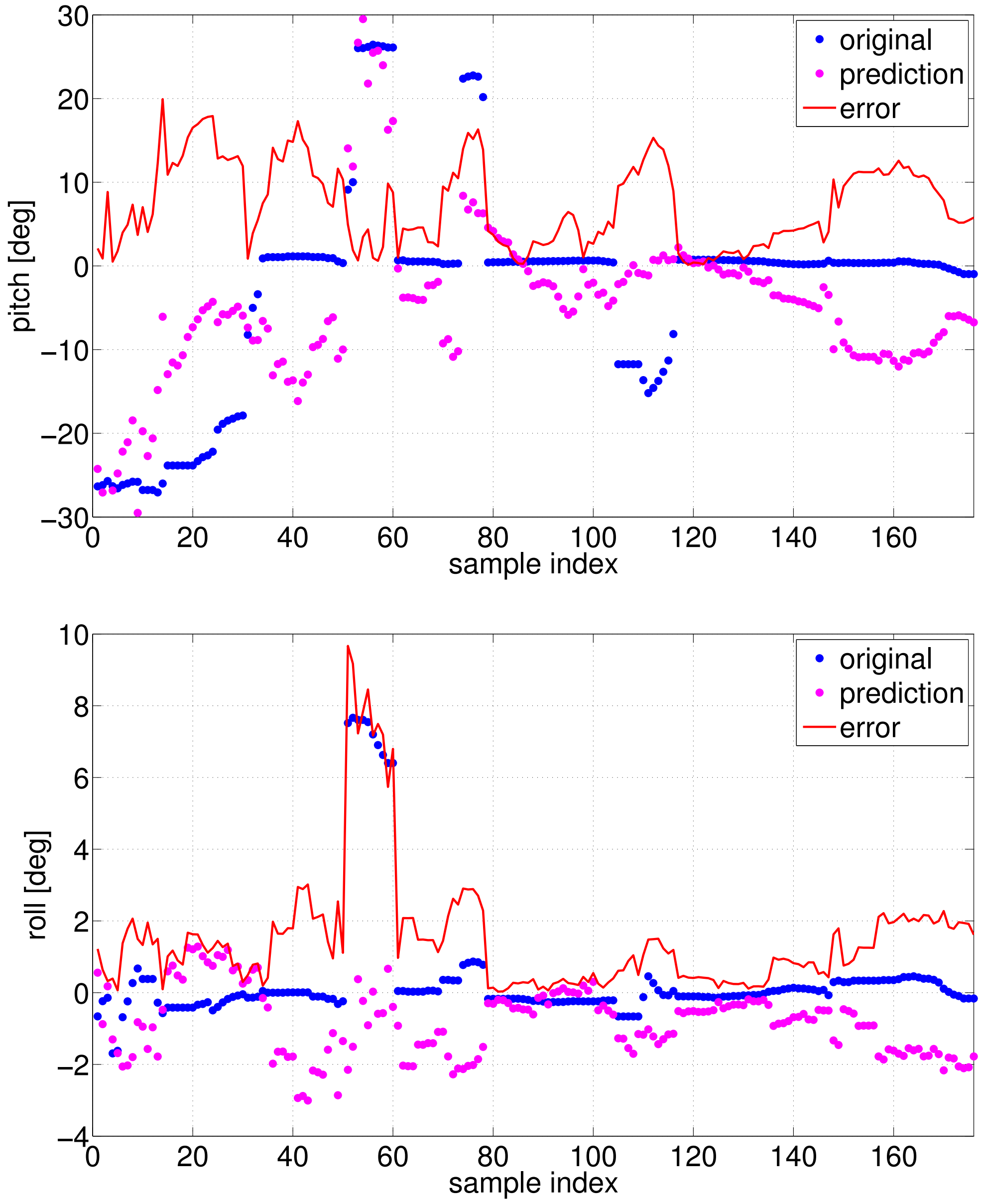


Figure 5.22: Linear regression for the 40 cm distance on the complex testing data.

5.3.2 PWC Regression

Fig. 5.23-5.30 show the PWC regression predictions for distances from 10 cm to 40 cm for simple and complex testing dataset for pitch and roll.

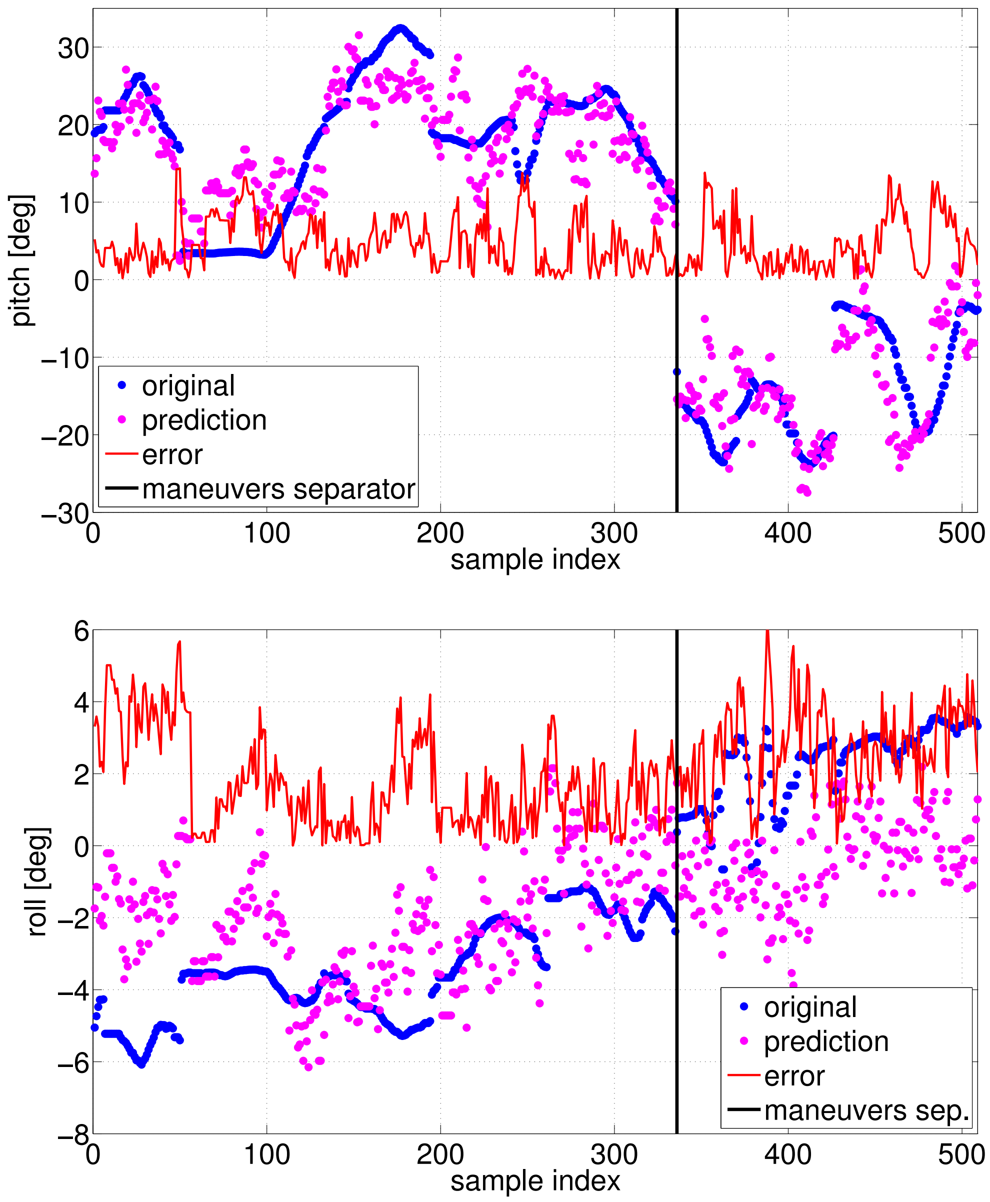


Figure 5.23: PWC regression for the 10 cm distance on the simple testing data.

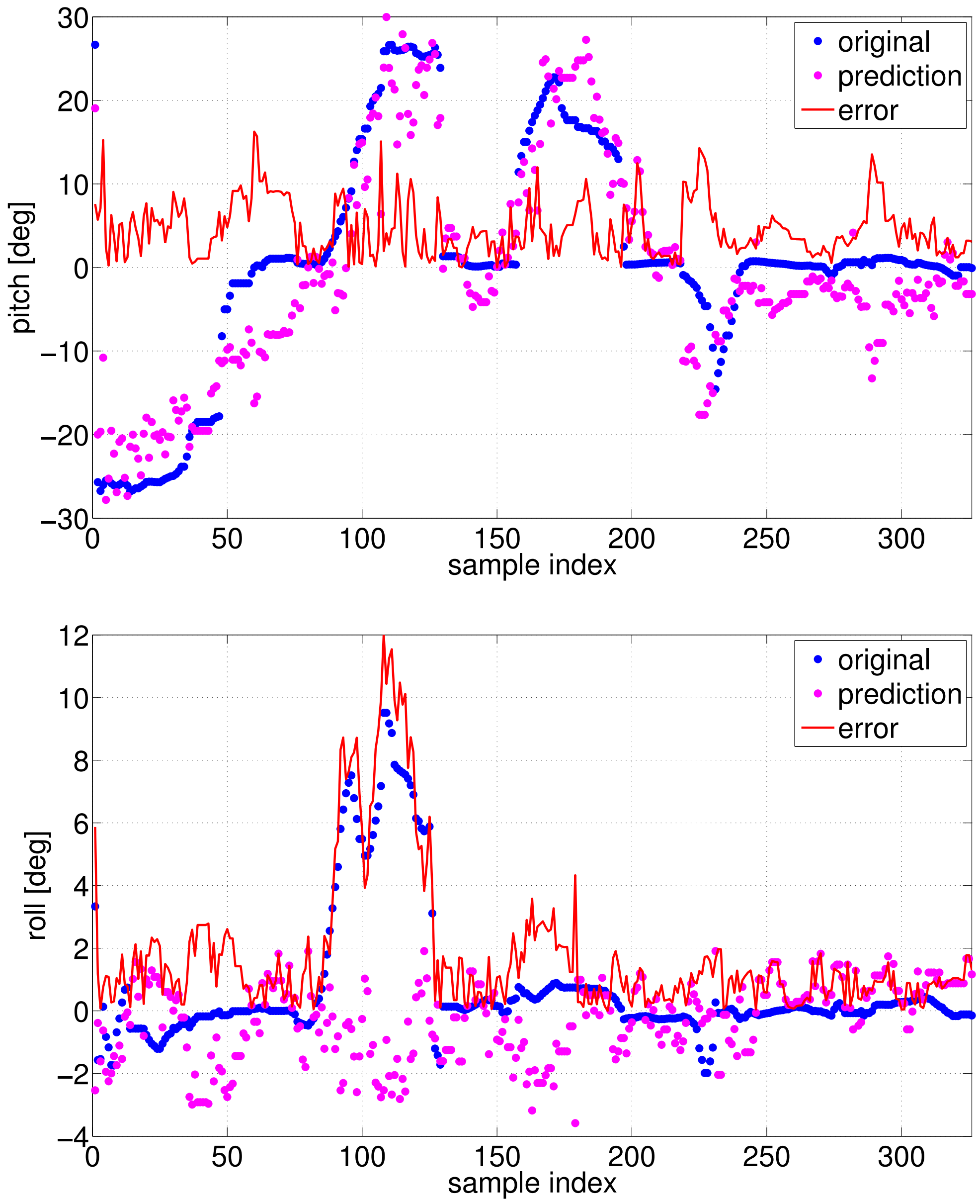


Figure 5.24: PWC regression for the 10 cm distance on the complex testing data.

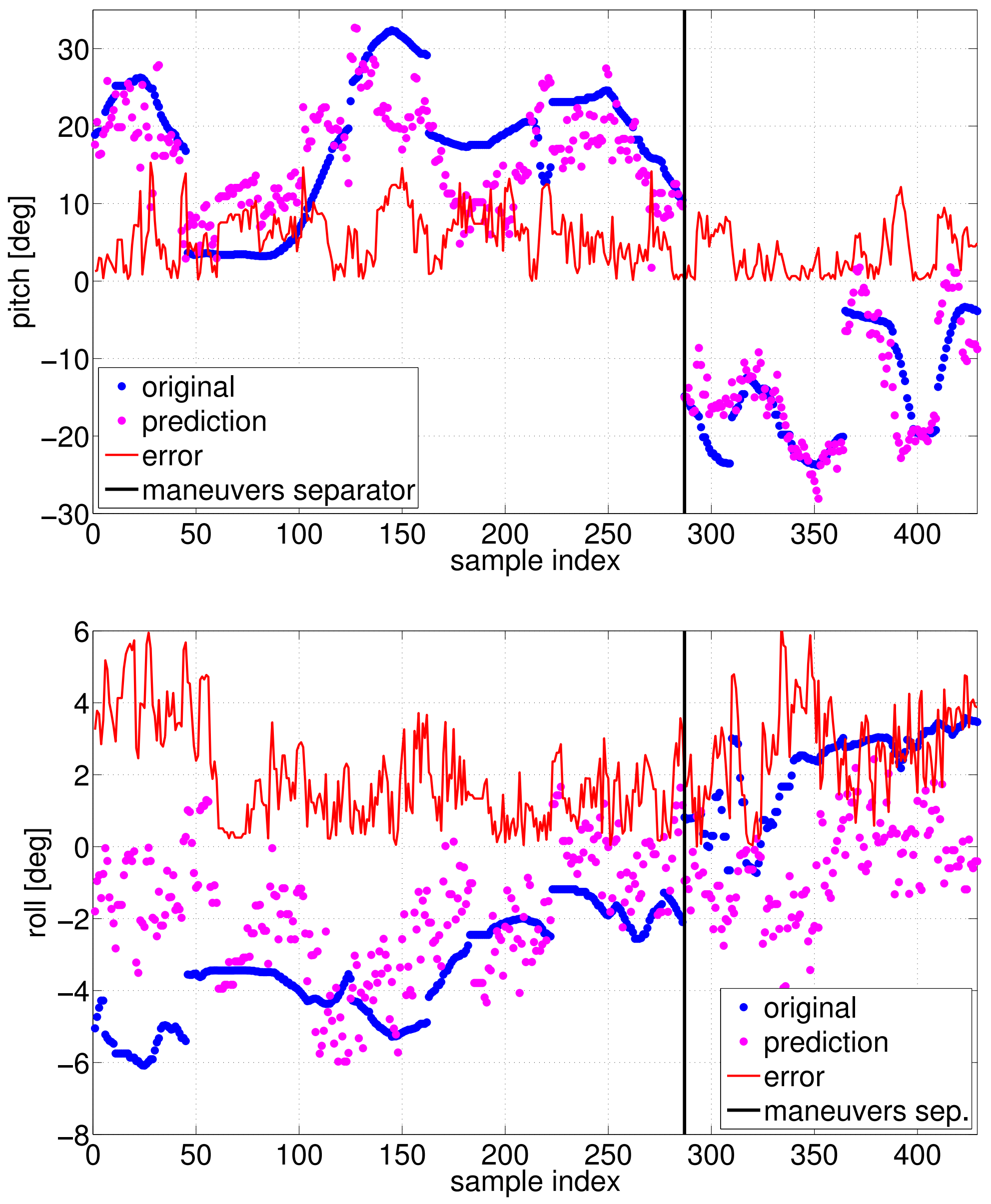


Figure 5.25: PWC regression for the 20 cm distance on the simple testing data.

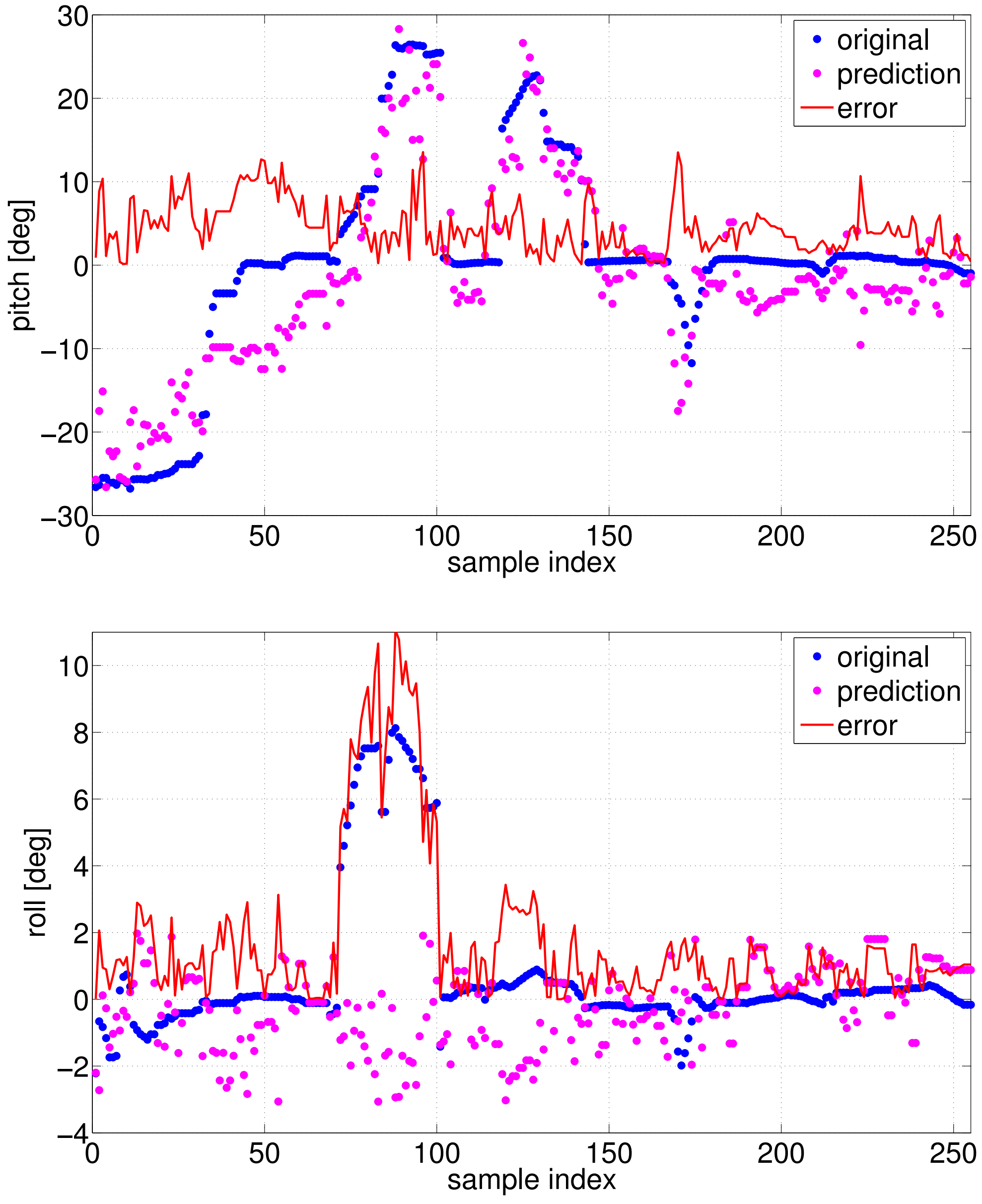


Figure 5.26: PWC regression for the 20 cm distance on the complex testing data.

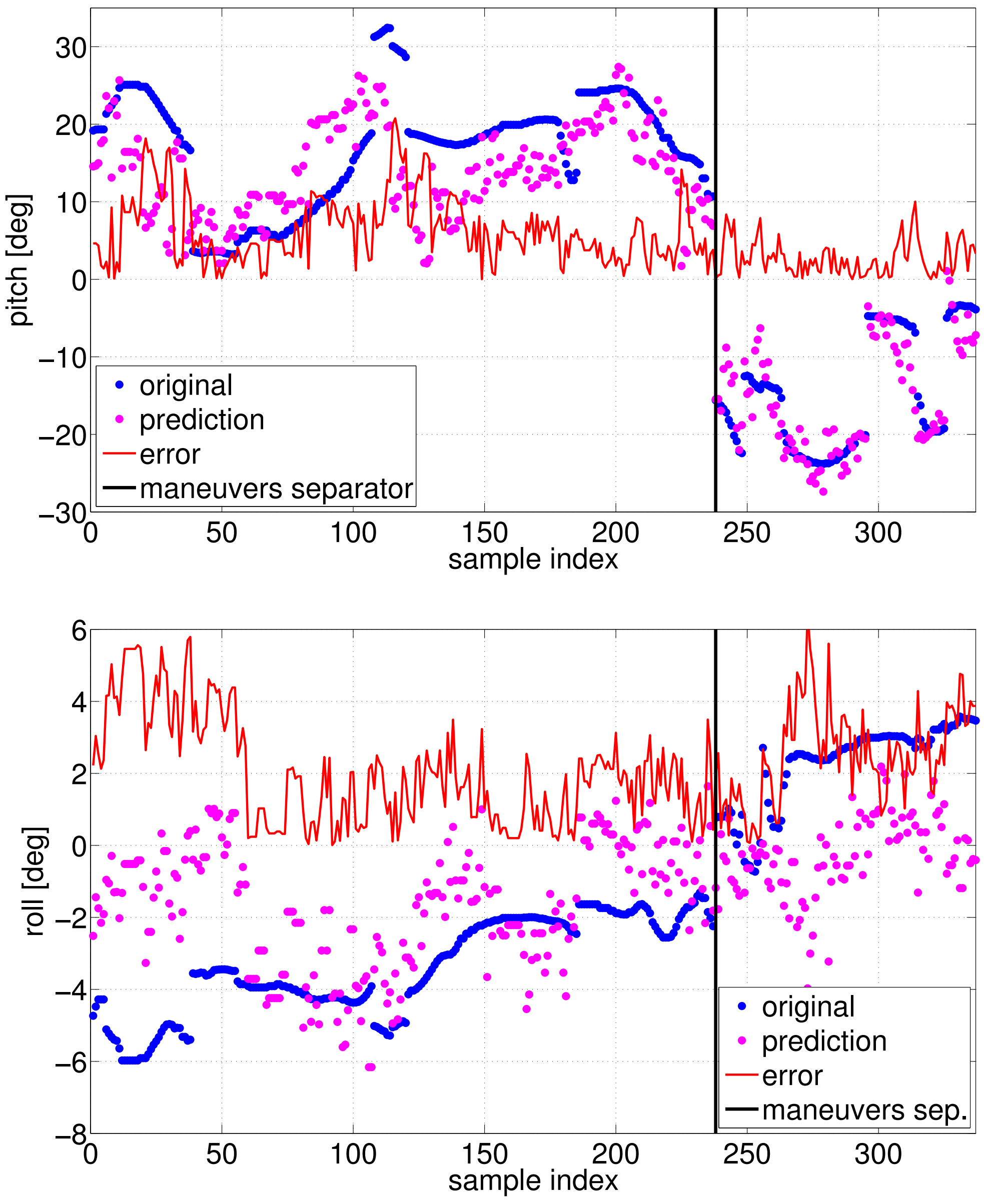
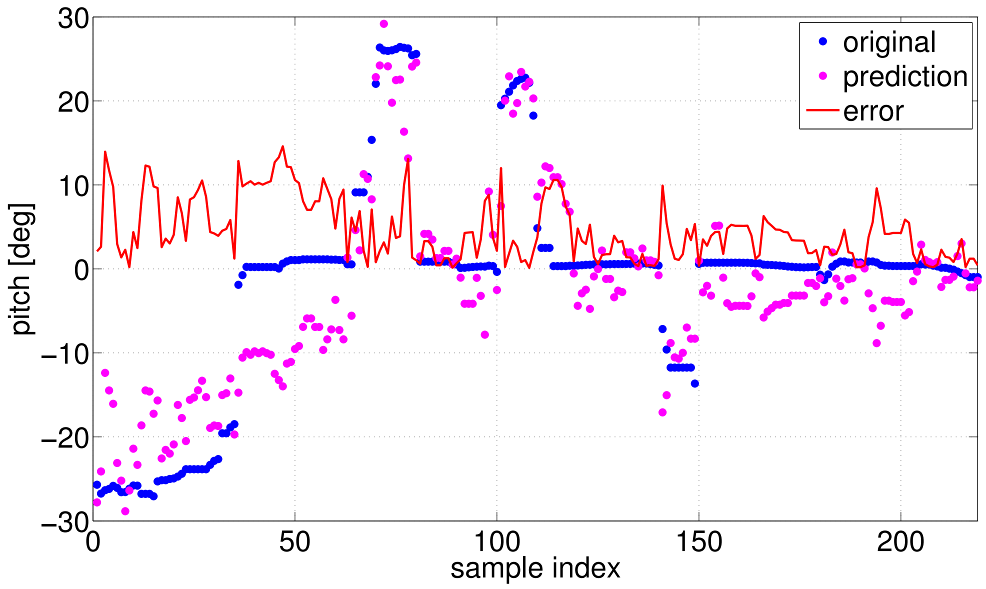


Figure 5.27: PWC regression for the 30 cm distance on the simple testing data.



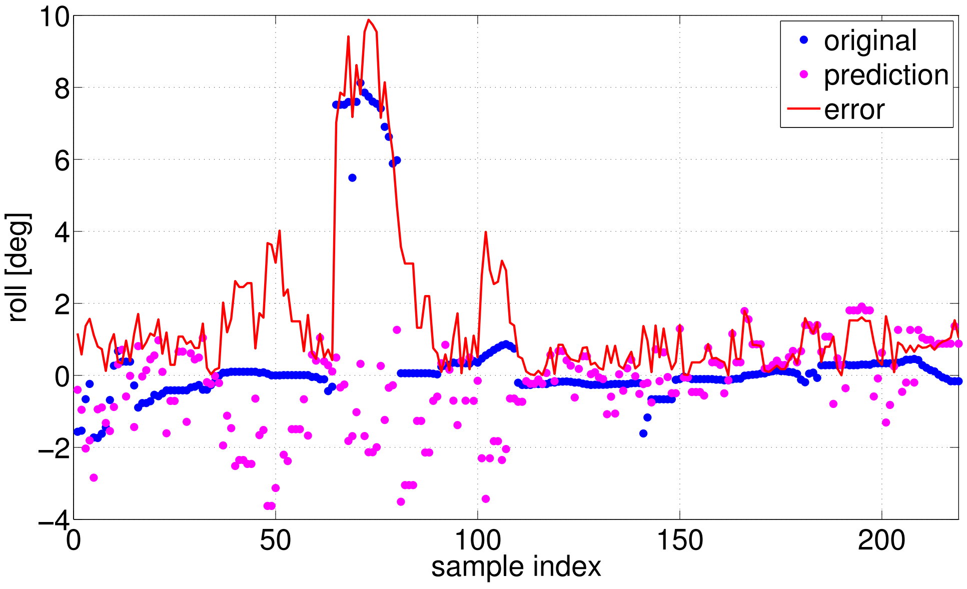
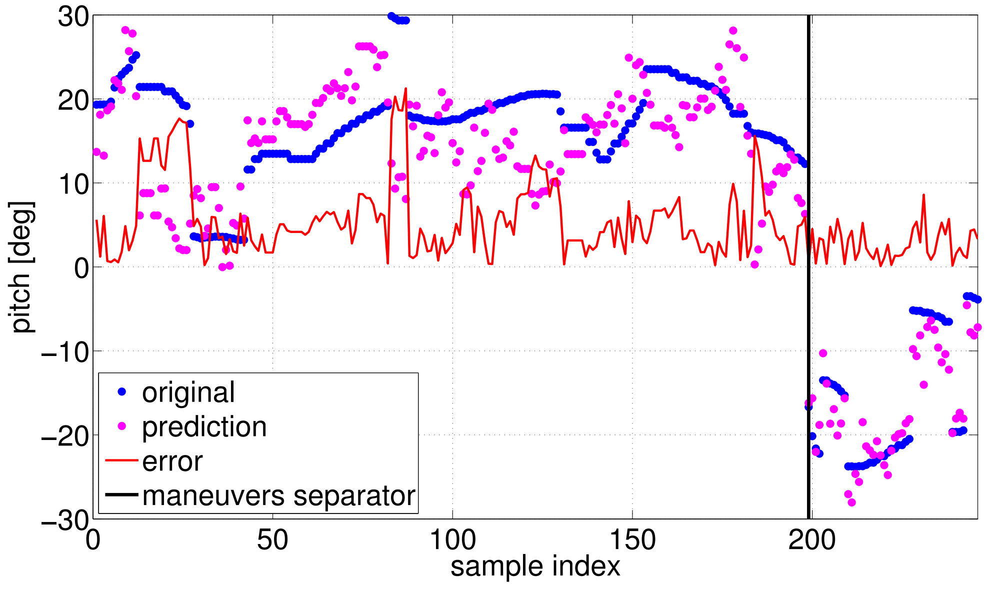


Figure 5.28: PWC regression for the 30 cm distance on the complex testing data.



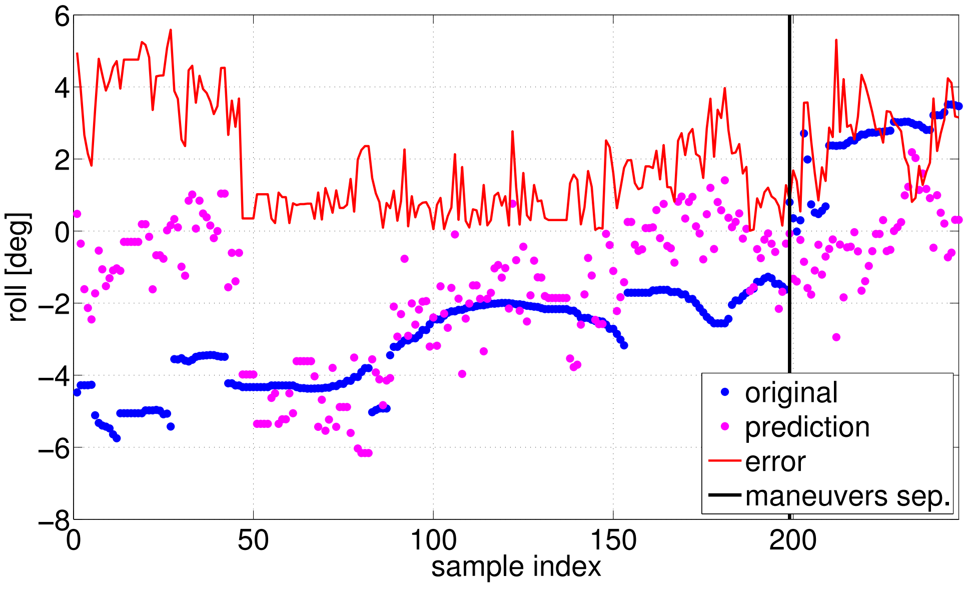


Figure 5.29: PWC regression for the 40 cm distance on the simple testing data.

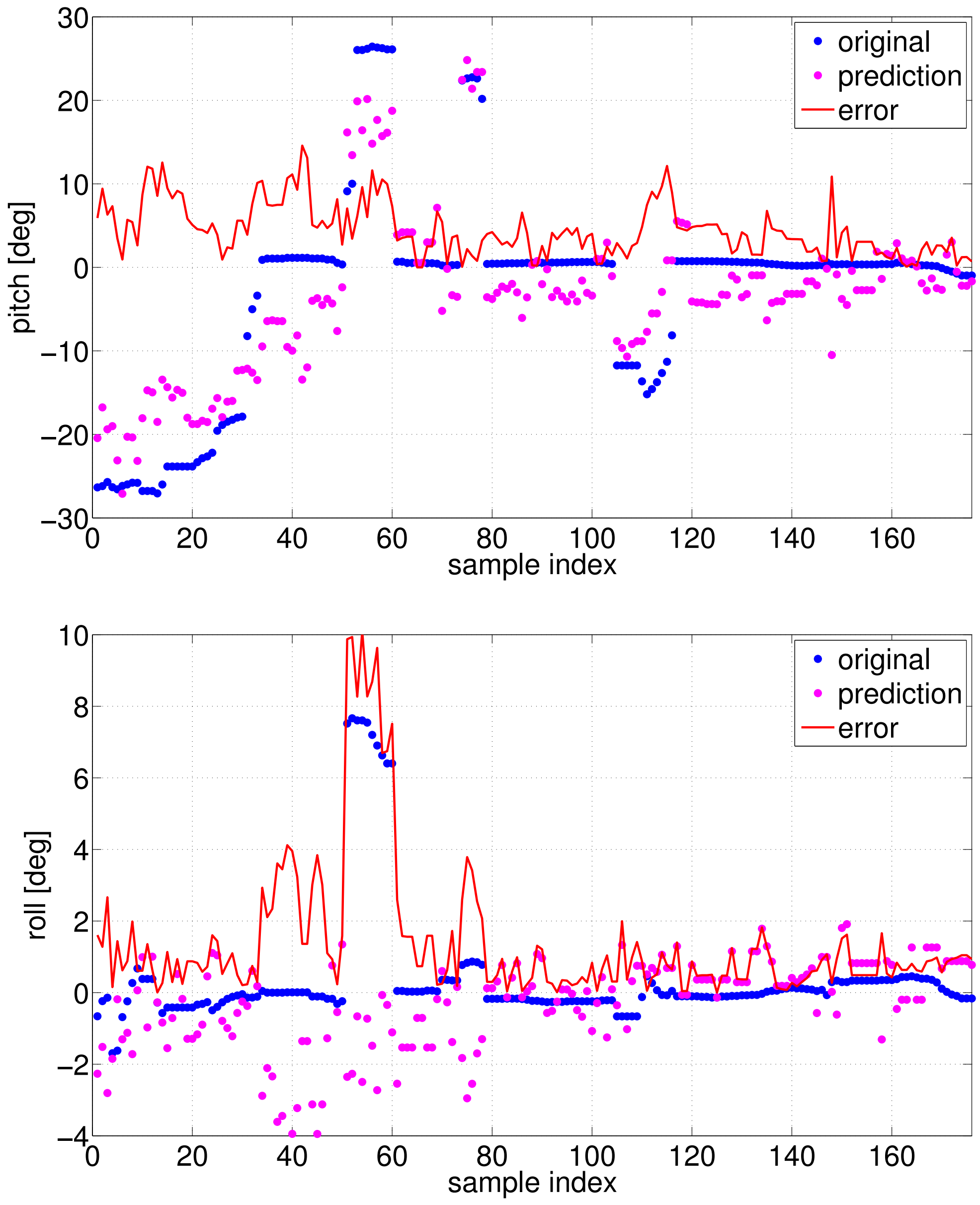


Figure 5.30: PWC regression for the 40 cm distance on the complex testing data.

5.3.3 GP Regression

Fig. 5.31-5.38 show the GP regression predictions for distances from 10 cm to 40 cm for simple and complex testing dataset for pitch and roll.

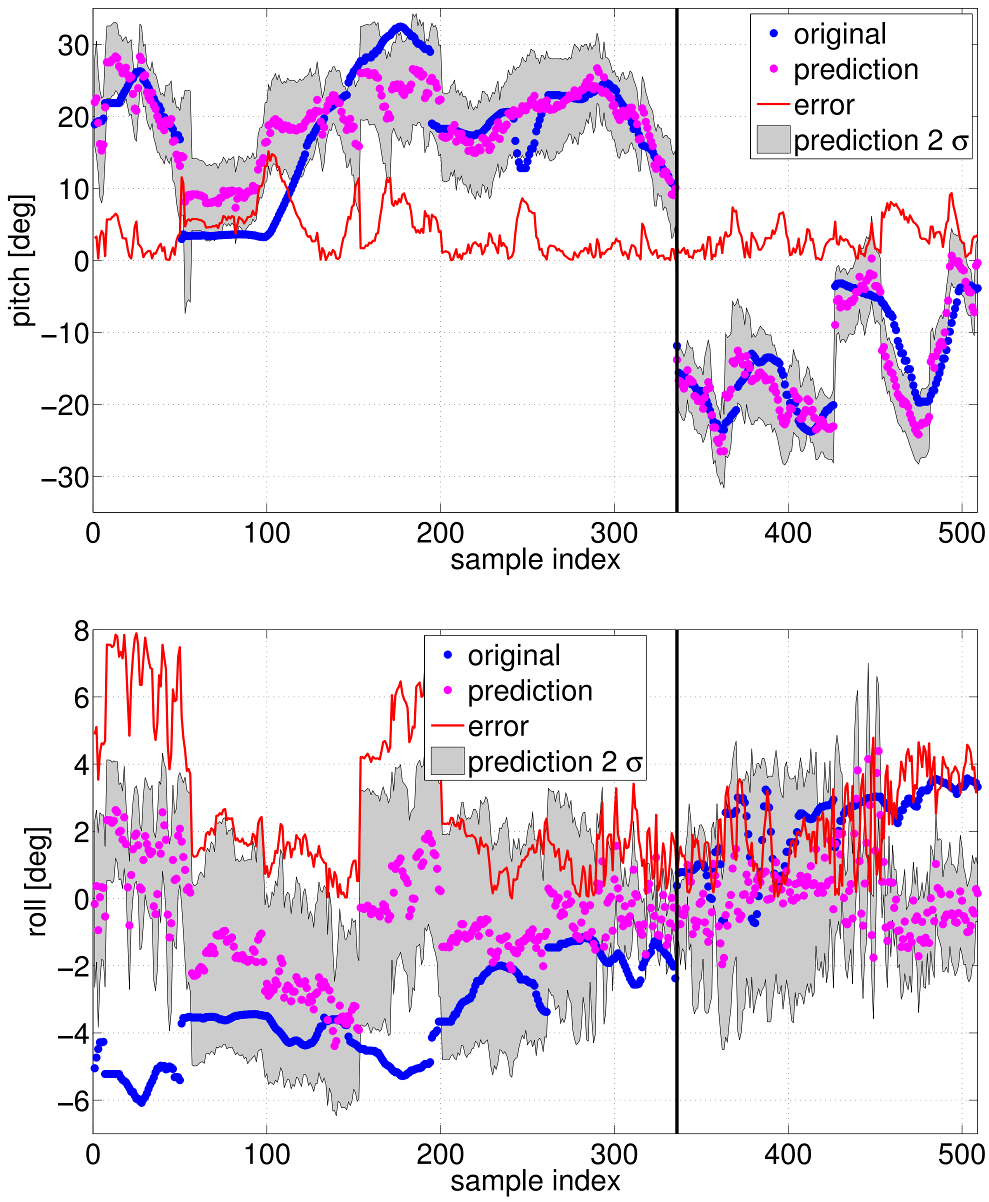


Figure 5.31: GP regression for the 10 cm distance on the simple testing data.

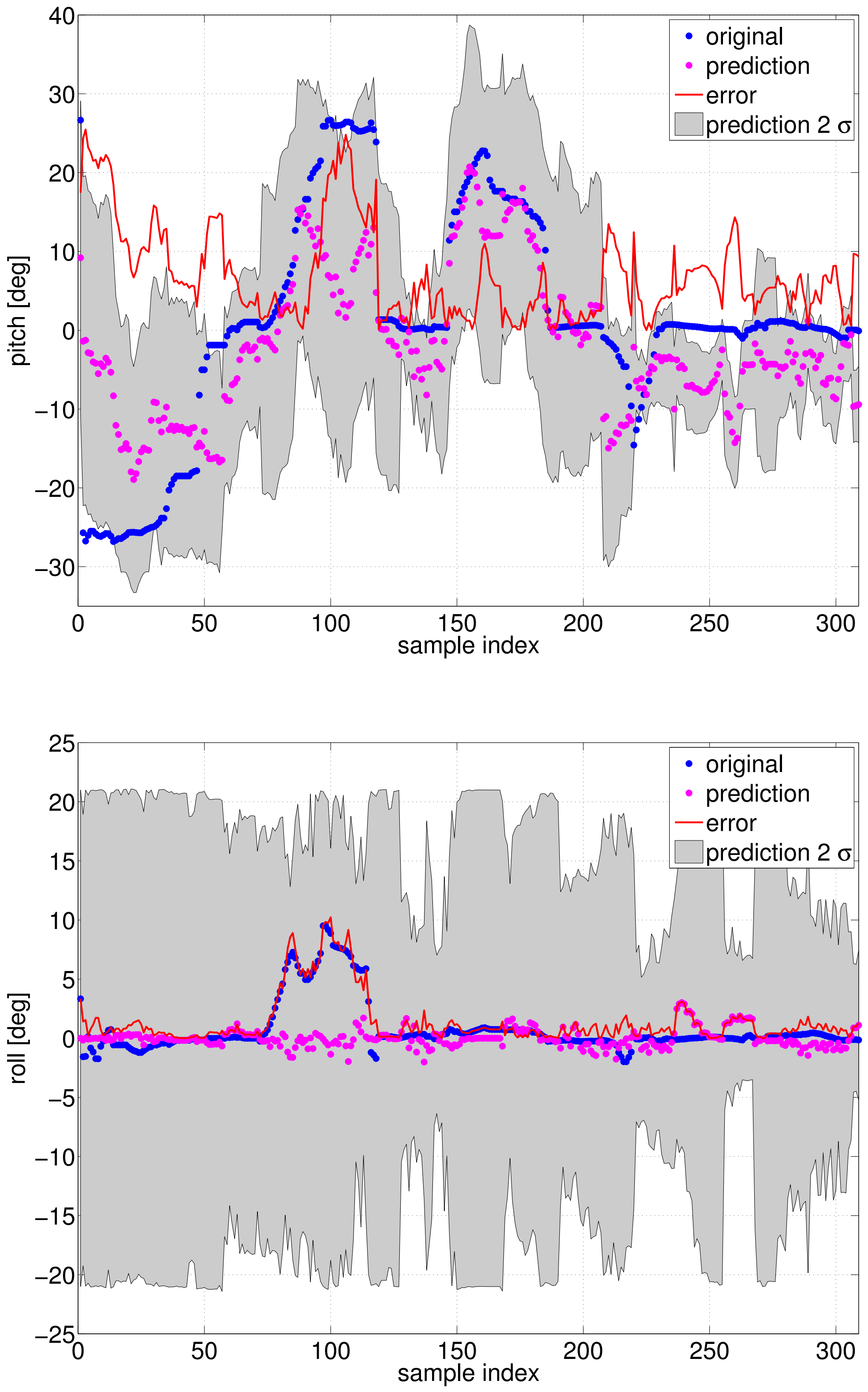


Figure 5.32: GP regression for the 10 cm distance on the complex testing data.

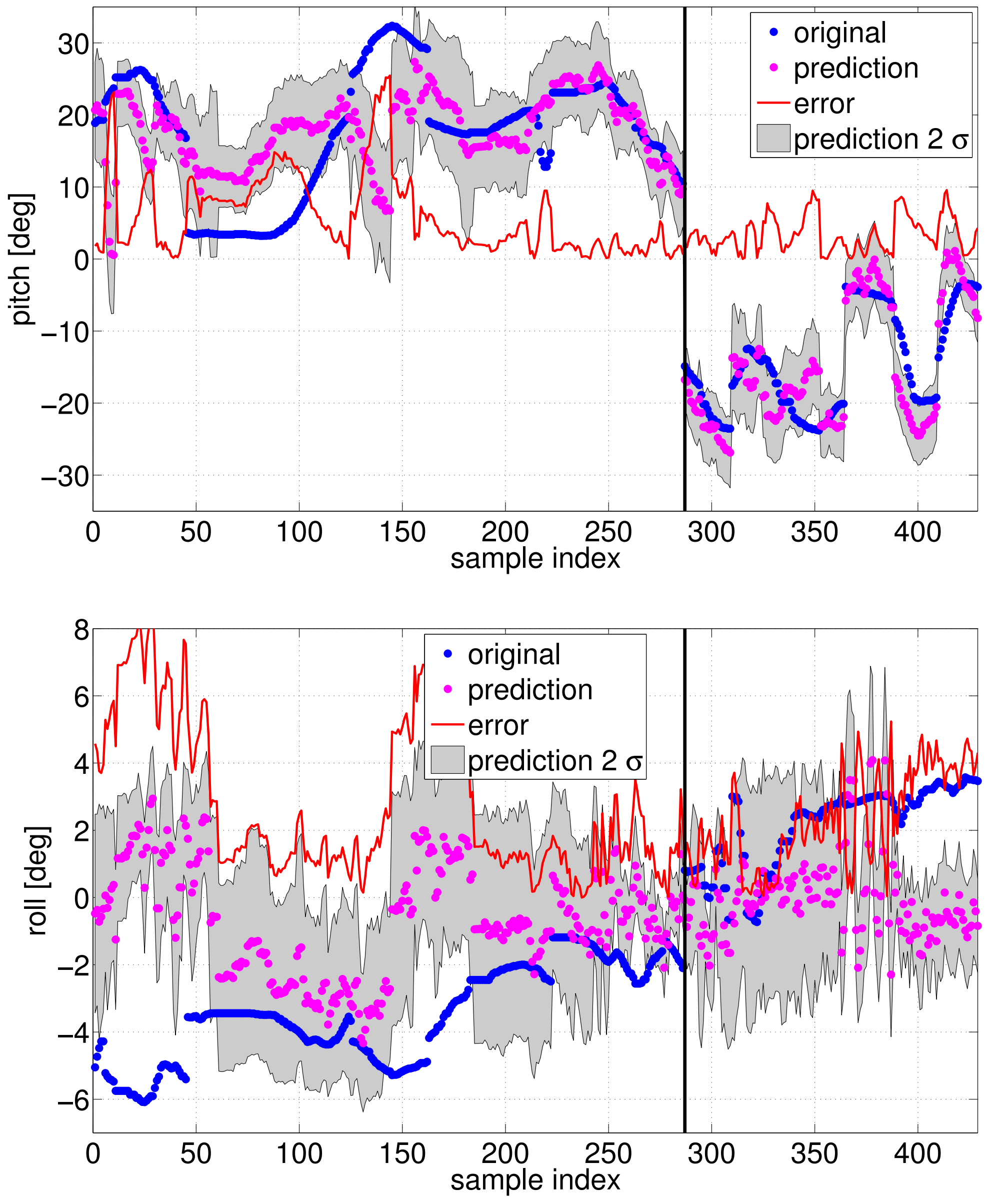
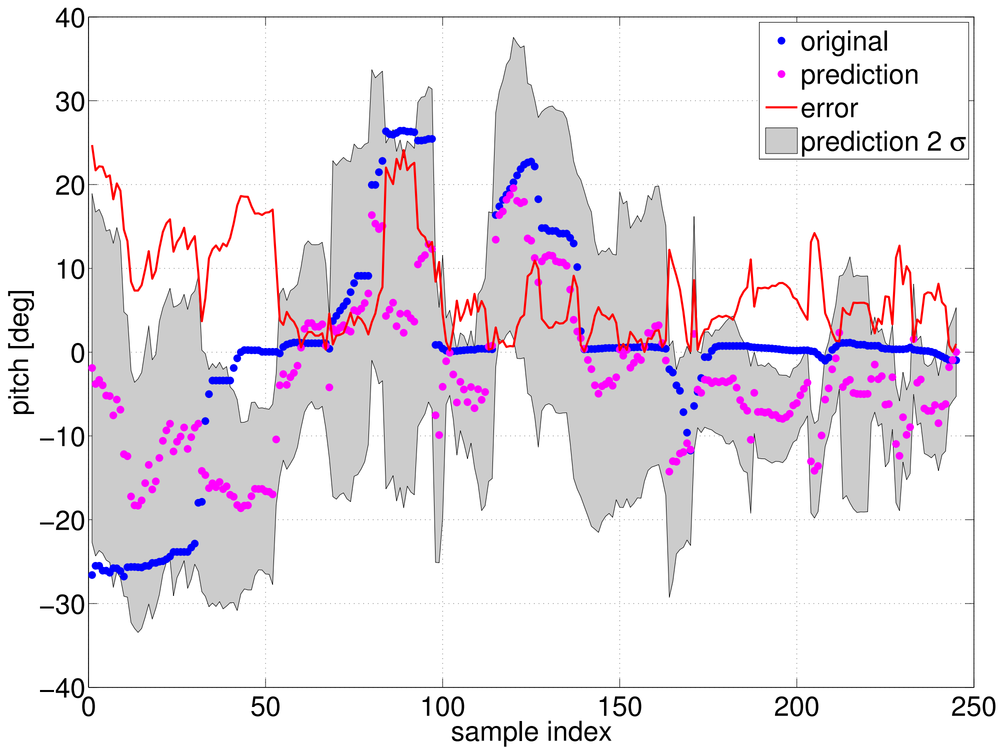


Figure 5.33: GP regression for the 20 cm distance on the simple testing data.



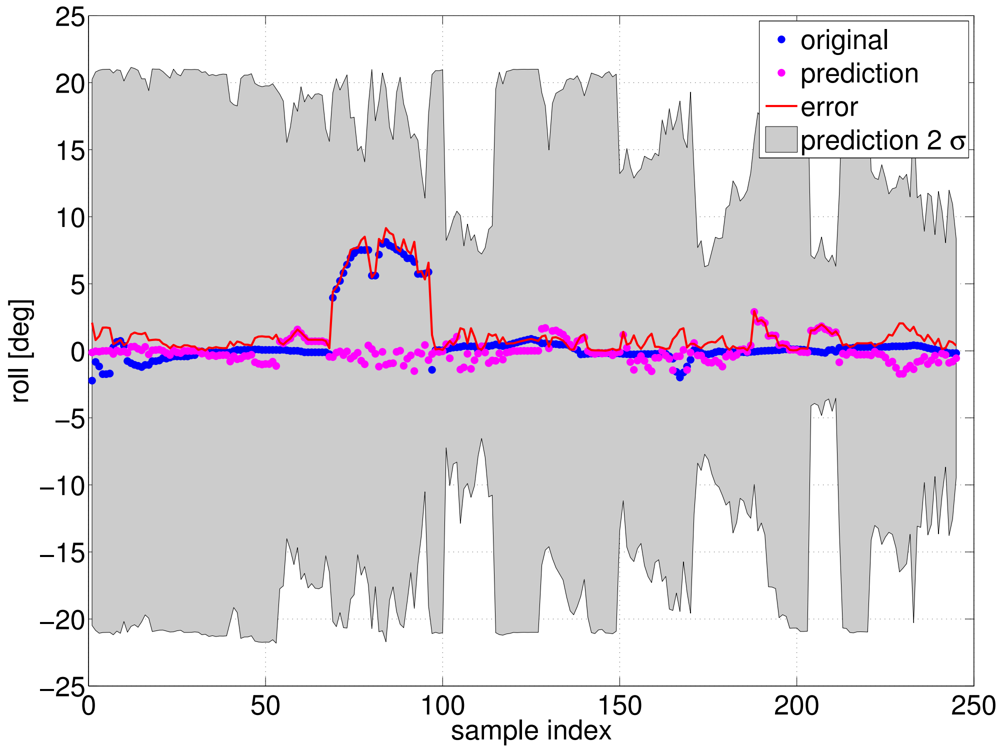
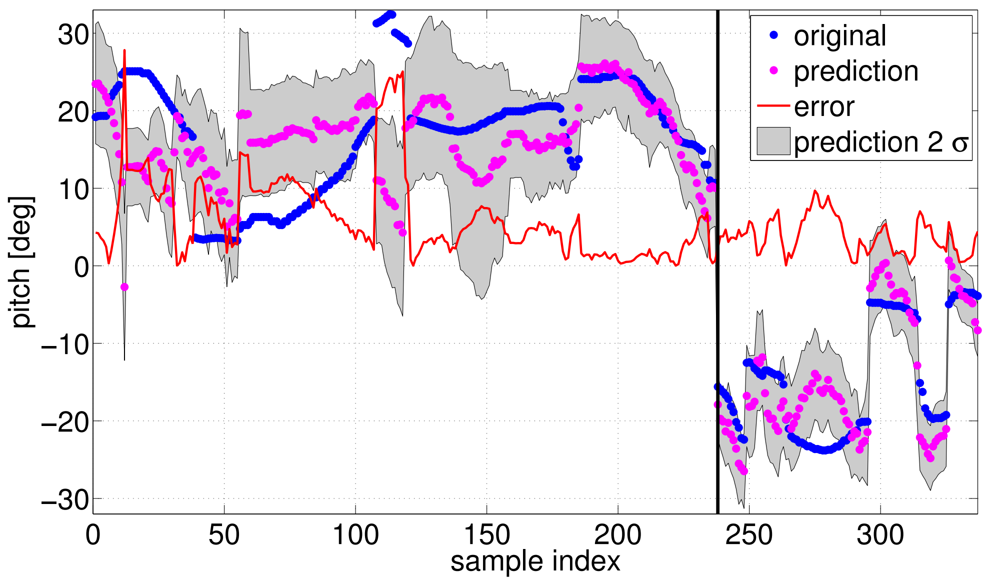


Figure 5.34: GP regression for the 20 cm distance on the complex testing data.



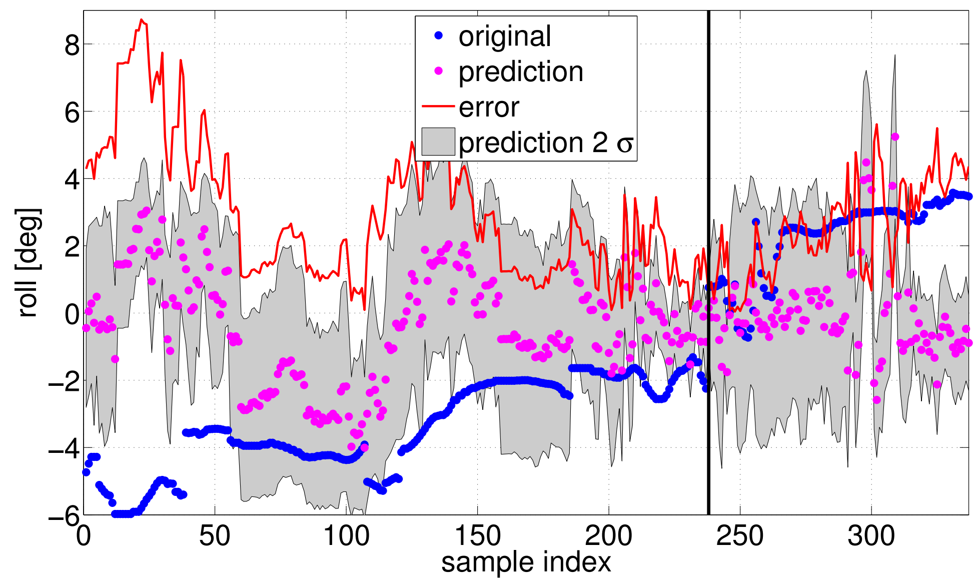
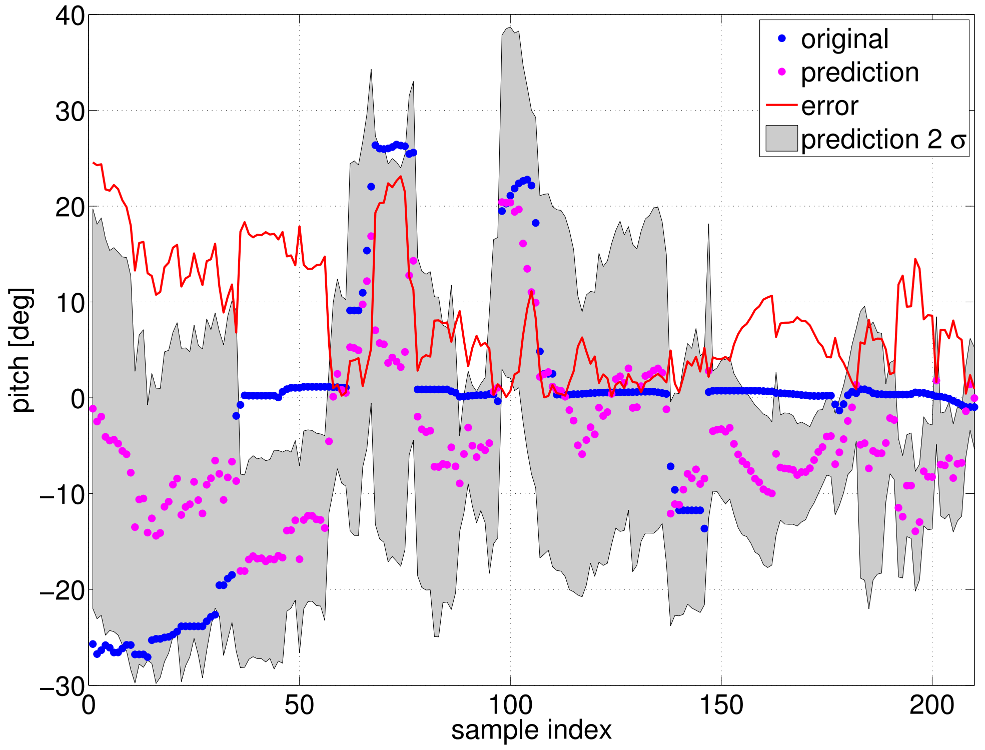


Figure 5.35: GP regression for the 30 cm distance on the simple testing data.



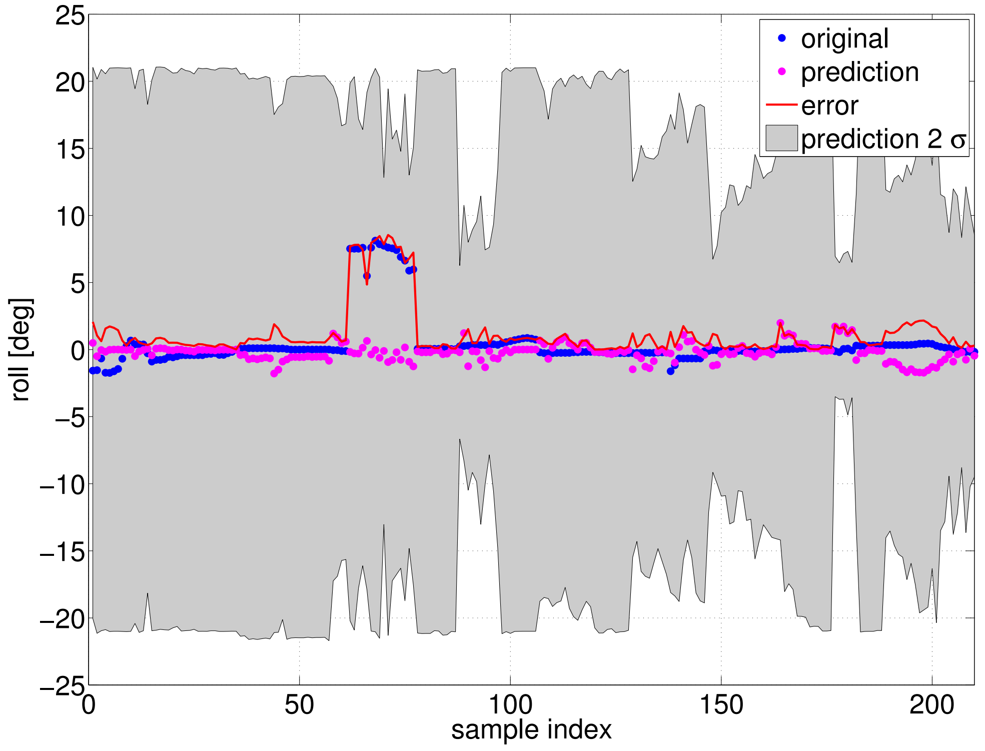


Figure 5.36: GP regression for the 30 cm distance on the complex testing data.

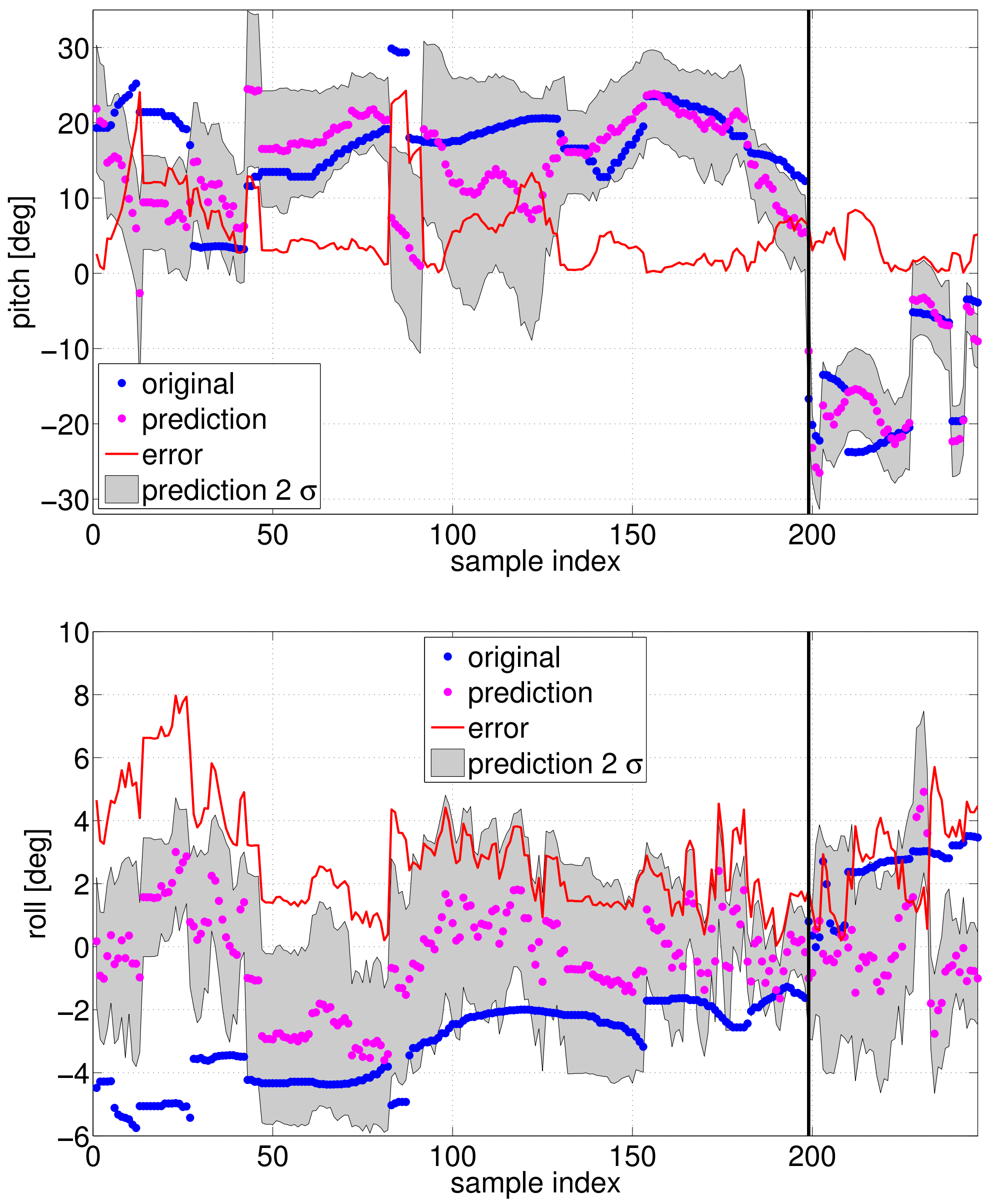
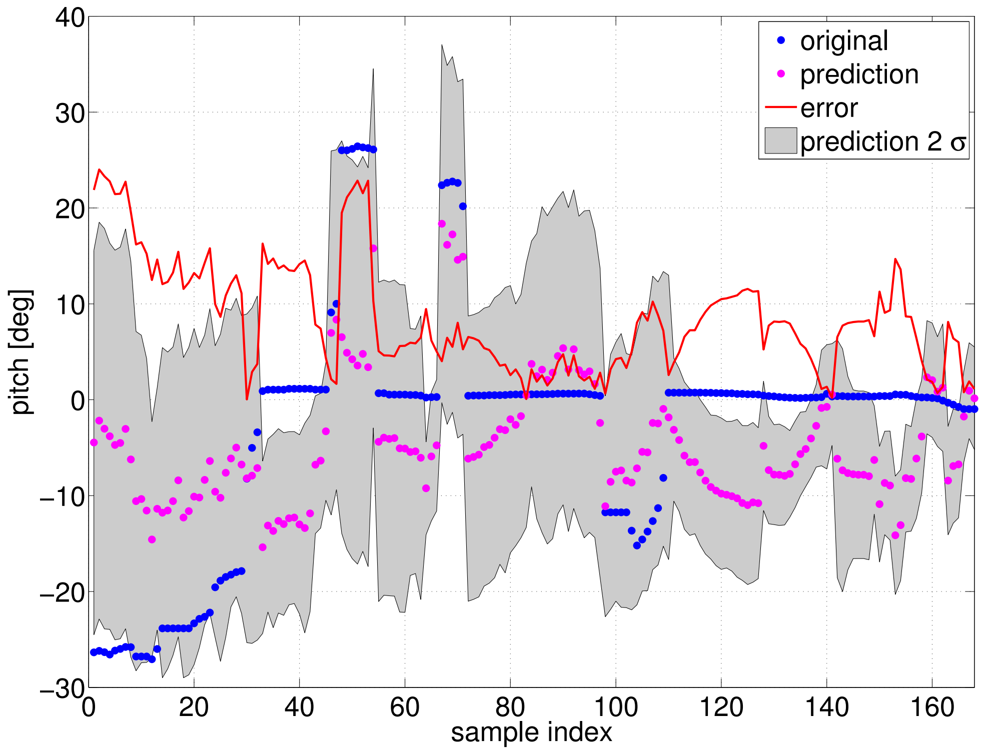


Figure 5.37: GP regression for the 40 cm distance on the simple testing data.



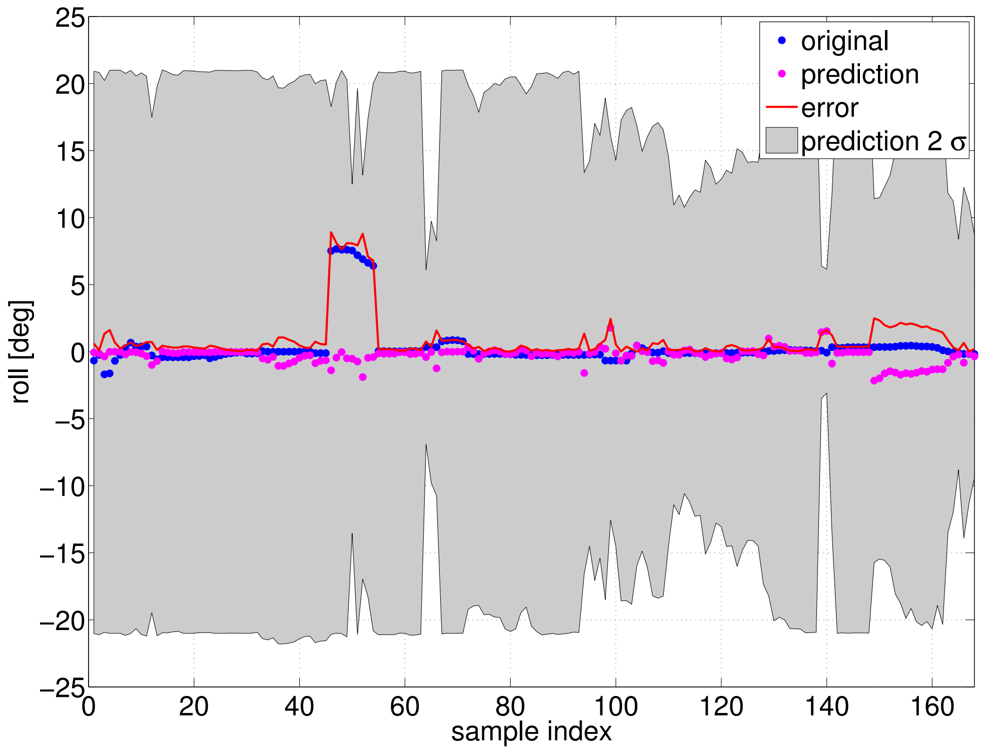


Figure 5.38: GP regression for the 40 cm distance on the complex testing data.

Table 5.1: Comparison of all regression methods for the pitch and the roll for the prediction distances up to 40 cm.

dataset	prediction distance [cm]	pitch RMS error [deg]			roll RMS error [deg]		
		linear	PWC	GP	linear	PWC	GP
training	0	8.75	5.95	0.99	1.87	1.50	0.04
simple	0	8.95	5.24	3.52	2.43	2.36	3.02
	10	8.59	5.68	4.71	2.60	2.47	3.23
	20	9.17	6.15	7.09	2.66	2.61	3.30
	30	9.63	6.77	7.40	2.71	2.60	3.39
	40	9.51	6.72	7.29	2.51	2.44	3.24
complex	0	7.04	5.45	10.26	2.82	3.07	2.57
	10	7.06	5.63	9.21	2.87	2.99	2.56
	20	7.41	5.41	9.66	2.88	2.96	2.58
	30	8.46	5.84	10.52	2.57	2.53	2.26
	40	8.84	5.51	10.27	2.28	2.43	2.00

6 Discussion

This chapter highlights some of the issues and unsolved problems, which are yet interesting but exceed the scope of this thesis.

6.1 ROS Implementation

The implementation of the proposed algorithm has not been deployed on the robot because the experiments have taken more time than expected and because a more complex ROS node for the autonomous robot safety is currently under development. It will contain other modules based on the reactive architecture as well as high level functionality, like presented in this thesis. The final specification was not established yet, so the C++ or Python implementation for ROS is left for the future work.

6.2 Wider DEM

The DEM used in this work had width of 50 cm, which covers the area between the flippers (see the Fig. 3.6). The reason is that the detection of points under flippers could be limited in some flipper modes and this absence of points should have an impact on the learning. The prediction might depend on the mode selected, instead of the real surface under flippers. This assumption was not tested primarily because the training samples used in this work contained mostly the horizontal obstacles wider than the robot, and the effect of DEM expansion to the sides should be minimal. However, in a future work it would be interesting to test the DEM widened to cover the width of the robot (60 cm), according to the Fig. 6.1, and see the consequences.

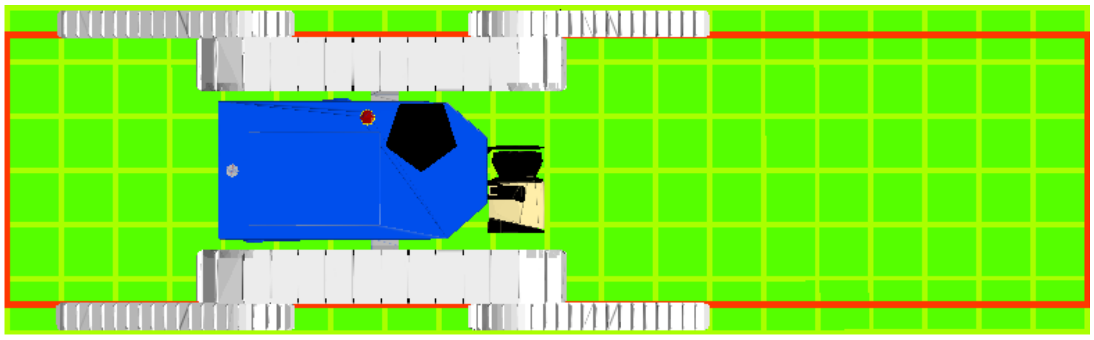


Figure 6.1: Robot on wider version of DEM in its default position. The original DEM size is indicated by a red rectangle.

6.3 Incomplete DEM

Big problem of the presented methods is the incomplete DEM. Although the missing cells values are computed from their neighborhood using a plane fitting, this approach can be used only for local defects. Current implementation is unable to distinguish between local occlusions (due to a small stone) and missing parts of the terrain (e.g. an edge of a cliff). For this reason a strict restriction were made in this work for the count of allowed missing cell values which leaded to a reduction of usability training set to about one third. Another approach was recently published by [31] where the GP reression is used to estimate the elevation of the DEM missing cells. Besides the nonlinear interpolation, GP also computes the uncertainty which can be used to distinguish between safe to interpolate areas and potentially dangerous holes in the terrain.

6.4 Limitation of the Model

Even though the proposed algorithm is mentioned to cooperate with a robot planner, it cannot be used to mark untraversable areas. Example situation is a steel rod sticking out of a concrete desk just before the robot. The desk is flat so it is suitable for safety traverse. A 3D sensor will detect the rod, it will be transferred to the DEM, but because the robot had never been standing on a tight tall obstacle before, it is eliminated as an outlier and the features describing the large flat area around will overweight it.

6.5 Configuration Mode Change Impact

The configuration mode has an impact on the resulting attitude angles, as shown in the Fig. 6.2 where the robot changes its pitch angle only by a change of the flippers configuration, while not moving. This angle change can be up to $8°$ on a same DEM and is the major source of the prediction errors. If this property is included into a feature vector, the predictors generalize from a fact, that before an obstacle, the robot operator often changed the configuration mode to one suitable for that kind of obstacle, but this is not usable for the general predictions, because other operator could change the mode in a different distance before the obstacle.

6.6 More Regression Types

In this work a PWC regression was used as a representative simple nonlinear regression method because there was a function tested implementation from a previous work. It would be interesting to compare its results with a similar method

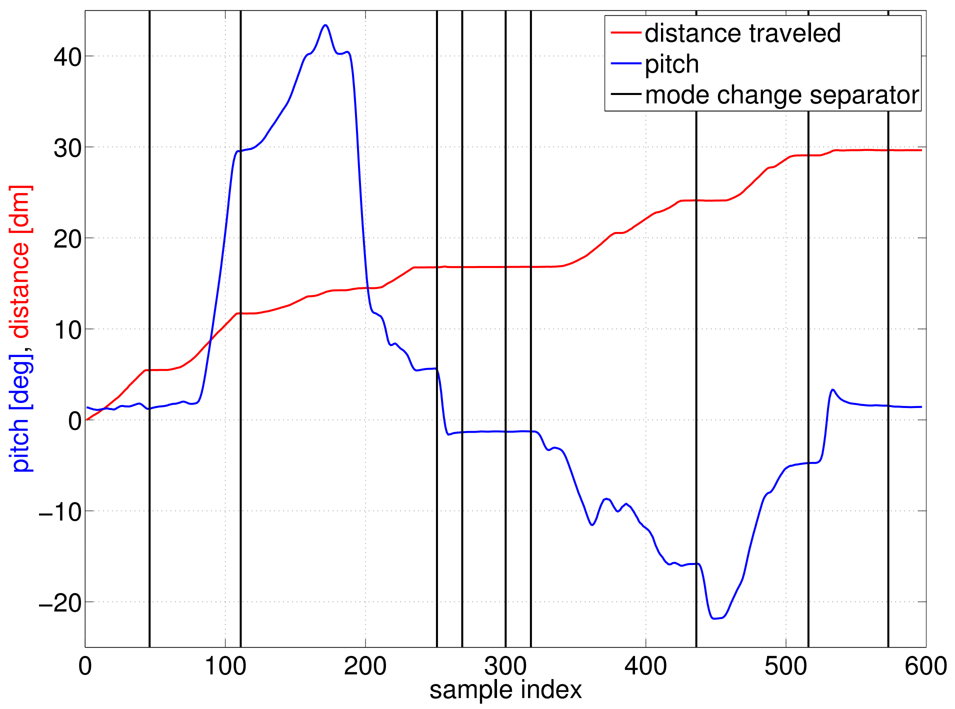


Figure 6.2: Dependency of the distance traveled and pitch angle on the sample index. On the index nr. 251 there is a change of 7° in pitch caused by a mode change, because the robot was not moving (red line is constant). Similar situation on the indexes nr.436 and 516.

called piecewise linear regression, which uses linear functions instead of constant values in joined regions.

Also, for the GP regression there are other kernel functions besides the used popular squared exponential, which would be interesting to test and compare (e.g. Gabor, Matérn, neural network, rational quadratic, etc.).

6.7 Prediction for Non-Straight Trajectory

Prediction method presented in this work assumes, that the robot will continue in a straight forward movement. For the real application, it would be interesting to extend this method to be able to predict the robot attitude according to an arbitrarily curved trajectory, that would be generated by a robot planner. This involves rotations of the DEM and its recalculations for discrete trajectory points. Prediction of the turning movement may be limited because of the sparser lidar detections on the sides of the robot, but it could be solved by, for example, slowing the rotation of the lidar (resulting in denser measurements but a slower response) or by using an additional 3D sensor mounted on the robot arm.

7 Conclusion

Three algorithms for predicting a robot attitude with respect to its stability, based on the machine learning methods, were proposed, trained and evaluated on different obstacles than used for training. Each algorithm consists of two models: one for predicting the pitch, second for the roll. Based on the results of the pitch predicting models, the linear model is not recommended for future usage. Results of the roll predicting models are not significant enough because the used training samples do not capture the whole range of motion dynamics.

Although the GP regression provided worse results in comparison to the PWC regression, it specifies prediction variance from which the precision of prediction can be stated. It also provides more stable and less noisy predictions and due to the automatic relevance detection no feature selection (and hence no cross-validation) is needed. While evaluating the GP method we discovered a great potential that can be exploited in the future work by expanding the training set and by examining more complex kernels. Regarding the accurancy of the IMU measurements, which is about one degree, the prediction error about $6°$ is acceptable.

Datasets used in this work aimed to cover maneuvers made by a human operator during crossing stair-like obstacles (palletes, concrete block, etc.). Although the training dataset contained about 3000 samples, it is insufficient to cover all the terrain characteristic (e.g. a forest terrain is completely different from the training samples) and further training will be necessary to improve the generalization of the predictor before deploying in the robot as a reliable application. Especially creation of dataset specialized on various roll angles is recommended, because it is not well covered in this work.

The proposed method aims to predict the safety of the robot in the meanings of stability by defining, that the robot is in danger of overturning, when its pitch angle is greater than $45°$ and the roll angle greater than $35°$. This is not sufficient to individually guarantee robot safety and assumes that it will be a part of a more complex system. Only predictions in the forward straight direction are computed. When the robot is, for example, on a narrow bridge, this method cannot ensure that the robot will be safe when turning around. It is also good to note that the behavior of the robot, when changing its configuration from one mode to another, is unknown.

All source codes and datasets created during this work are stored on the CMP [32] datagrid and after the end of the TRADR project will be opened to the worldwide robotic community.

Bibliography

[1] Karel Zimmermann, Petr Zuzanek, Michal Reinstein, and Vaclav Hlavac. Adaptive traversability of unknown complex terrain with obstacles for mobile robots. In *International Conference on Robotics and Automation (ICRA)*, 2014. accepted for publication.

[2] S. Singh and P. Keller. Obstacle detection for high speed autonomous navigation. In *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*, pages 2798–2805 vol.3, Apr 1991. doi:10.1109/ROBOT.1991.132057.

[3] Zhongfei Zhang, R. Weiss, and A.R. Hanson. Qualitative obstacle detection. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 554–559, Jun 1994. doi:10.1109/CVPR.1994.323881.

[4] S. Badal, S. Ravela, B. Draper, and A. Hanson. A practical obstacle detection and avoidance system. In *Applications of Computer Vision, 1994., Proceedings of the Second IEEE Workshop on*, pages 97–104, Dec 1994. doi:10.1109/ACV.1994.341294.

[5] T. Williamson and C. Thorpe. A specialized multibaseline stereo technique for obstacle detection. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, pages 238–244, Jun 1998. doi:10.1109/CVPR.1998.698615.

[6] Natural human-robot cooperation in dynamic environments. Accessed: 25/05/2012. Available from: www.nifti.eu.

[7] Long-term human-robot teaming for robot assisted disaster response. Accessed: 07/04/2014. Available from: http://www.tradr-project.eu/.

[8] Darpa robotic challenge. Accessed: 29/04/2014. Available from: http://www.theroboticschallenge.org/.

[9] Simon Lacroix, Anthony Mallet, David Bonnafous, Gérard Bauzil, Sara Fleury, Matthieu Herrb, and Raja Chatila. Autonomous rover navigation on unknown terrains: Functions and integration. *International Journal of Robotics Research*, 21:2002, 2002.

[10] Alberto Lacaze, Karl Murphy, and Mark DelGiorno. Autonomous mobility for the demo iii experimental unmanned vehicles. In *in Assoc. for Unmanned Vehicle Systems Int. Conf. on Unmanned Vehicles (AUVSI 02*, 2002.

[11] Mahmoud Tarokh and Gregory J. McDermott. Kinematics modeling and analyses of articulated rovers. *IEEE Transactions on Robotics*, 21:539–553, 2005.

[12] Ken Ho, Thierry Peynot, and Salah Sukkarich. Traversability estimation for a planetary rover via experimental kernel learning in a gaussian process framework. In *Internation Conference on Robotics and Automation (ICRA)*, 2013.

[13] Ken Ho, T. Peynot, and S. Sukkarieh. A near-to-far non-parametric learning approach for estimating traversability in deformable terrain. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 2827–2833, Nov 2013. doi:10.1109/IROS.2013.6696756.

[14] Absolem - Surveillance & Rescue. Accessed: 08/05/2014. Available from: http://www.bluebotics.com/mobile-robotics/absolem/.

[15] Robot Operating System. Accessed: 16/01/2012. Available from: http://www.ros.org/wiki/ROS.

[16] Open Source Robotics Foundation. Accessed: 08/05/2012. Available from: http://www.osrfoundation.org/.

[17] ROS node. Accessed: 27/01/2012. Available from: http://www.ros.org/wiki/Nodes.

[18] ROS topic. Accessed: 27/01/2012. Available from: http://www.ros.org/wiki/Topics.

[19] octomap_server node. Accessed: 01/05/2014. Available from: http://wiki.ros.org/octomap_server.

[20] OctoMap library. Accessed: 01/05/2014. Available from: http://octomap.github.io/.

[21] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 2013. Software available at http://octomap.github.com. Available from: http://octomap.github.com, doi:10.1007/s10514-012-9321-0.

[22] V. Kubelka and M. Reinstein. Complementary filtering approach to orientation estimation using inertial sensors only. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 599–605, May 2012. doi:10.1109/ICRA.2012.6224564.

[23] Documentation for GPML Matlab Code version 3.4. Accessed: 10/05/2014. Available from: http://www.gaussianprocess.org/gpml/code/matlab/doc/.

[24] Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

[25] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. doi:10.1109/CVPR.2001.990517.

[26] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. Available from: http://dx.doi.org/10.1016/j.cviu.2007.09.014, doi:10.1016/j.cviu.2007.09.014.

[27] Andrew E. Yagle. Regularized Matrix Computations. Accessed: 11/04/2014. Available from: web.eecs.umich.edu/~aey/recent/regular.pdf.

[28] A tutorial on gaussian processes (or why i don't use svms). Accessed: 2/05/2014. Available from: mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf.

[29] MATLAB (matrix laboratory). Accessed: 22/04/2014. Available from: http://www.mathworks.com/products/matlab/.

[30] rosbag. Accessed: 22/04/2014. Available from: http://ros.org/wiki/rosbag.

[31] Ken Ho, Thierry Peynot, and Salah Sukkarieh. Analysis of terrain geometry representations for traversability of a mars rover. In *11th Australian Space Science Conference*, pages 359–372, Canberra, Australia, September 2011. National Space Society of Australia Ltd. Available from: http://eprints.qut.edu.au/67613/.

[32] Center for Machine Perception at Czech Technical University in Prague. Accessed: 04/05/2014. Available from: http://cmp.felk.cvut.cz.

CD content

Attached CD contains a text of this master thesis in the PDF format and source codes of the whole text for the LaTeX.

The directory tree is in the next table:

Table 1: Directory tree of the attached CD

Directory	Label
`LaTeX`	source codes for the text of the thesis
↪ `src`	directory containing individual chapters
↪ `fig`	directory with used images
↪ `plots`	directory with used plots
↪ `Makefile`	Makefile for building the thesis
↪ `thesisPrint.pdf`	printable version of the thesis
`thesis.pdf`	CD version of the thesis

List of Figures