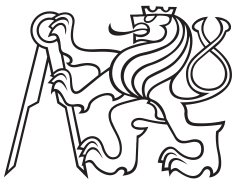
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Surface Properties Prediction from Images Using Self-supervised Learning

Bc. Adam Konopiský

Supervisor: Ing. Jan Čech, Ph.D.
Field of study: Cybernetics and Robotics
May 2022

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Konopiský Adam** Personal ID number: **474448**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Surface Properties Prediction from Images Using Self-Supervised Learning

Master's thesis title in Czech:

Predikce vlastností povrchu z obrazu s využitím samoučení

Guidelines:

In recent years, we developed a method for a visual prediction of the drivable surface properties in front of a vehicle, e.g., surface roughness and friction [1]. The main principle is that a test vehicle is freely driving on various surfaces while recording images from a front-looking camera and data from several nonvisual sensors (accelerometers, wheel RPMs, motor torque, etc.) The surface properties are derived from the vehicle response to the surface, using the nonvisual sensors. The surface properties measured this way then serve as a label and are associated with the corresponding images captured by the camera. Convolutional neural regressor is trained to predict the surface property label. A weakness of the methodology is that the regressor predicts a single property for an entire image and lacks more detailed spatial resolution. This is particularly important for navigating on complex heterogeneous surfaces comprising various areas of different properties. The problem is to increase the spatial resolution of the predictor. The output will be a map of local surface properties.

1. Make a survey of related literature.
2. Propose a method for a prediction of local maps of surface properties using weak supervision.
3. Perform experiments on data from the subscale vehicle platform [1].
4. Evaluate the method quantitatively .
5. (Optionally) test the algorithm in the autonomous pipeline, when the experimental vehicle should follow a given surface.

Bibliography / sources:

[1] Cech et al., Self-Supervised Learning of Camera-based Drivable Surface Roughness. In Proc. IEEE Intelligent Vehicles Symposium, 2021.
[2] Selvaraju et al., Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, ICCV 2017.
[3] Zhu et al., Learning Instance Activation Maps for Weakly Supervised Instance Segmentation, CVPR 2019.

Name and workplace of master's thesis supervisor:

Ing. Jan Čech, Ph.D. Visual Recognition Group FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **27.01.2022** Deadline for master's thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Jan Čech, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor's signature	Head of department's signature	Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to thank my supervisor Ing. Jan Čech, PhD. for his excellent guidance. I would also like to thank doc.Ing. Tomáš Haniš, PhD. and Ing. David Vošahlík for their expertise consultations.

My thanks also belong to members of the Tomi2 project: Tomáš Twardzik, Marek Boháč, Jan Švancar and Tomáš Rutrle. This thesis is a result of our cooperation, for which I am really thankful.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 2022.

.....
Adam Konopiský

Abstract

This thesis develops techniques for predicting a map of local surface properties in front of a vehicle. Traditional automotive control systems adapt to road conditions reactively. Instead, we propose to adjust their parameters based on a surface property prediction, before the vehicle travels over the surface, which can potentially improve safety, comfort, and maintenance costs of modern cars.

Two surface properties are used: roughness and friction. The dataset is obtained in a self-supervised fashion, meaning that there is no manual annotation. When training, the image labels are obtained automatically by properties derived from vehicle responses (vertical acceleration, slip ratio, torque etc.). When the system is deployed, only the images are used to predict the surface properties.

This thesis builds on top of the results achieved in our previous work (Cech et al. 2021[1] and Vosahlik et al. 2021 [2]). Previous models were limited to predicting a single surface property for an entire image. We propose two approaches. The first uses U-Net convolutional neural network to output a pixel-wise property map trained on the semi-synthetic dataset. The second method uses visual explanations from deep networks via gradient-based localization (Grad-CAM). All experiments were done using the subscale vehicle platform. Both models were validated on a manually annotated test dataset, showing promising results.

Keywords: surface properties, convolutional neural networks, automatic annotation

Supervisor: Ing. Jan Čech, Ph.D.

Abstrakt

Tato práce se zaměřuje na vývoj technik pro predikci mapy povrchových vlastností před vozidlem. Tradiční řídicí systémy v automobilovém průmyslu se přizpůsobují podmínkám vozovky reaktivním způsobem, takže schopnost upravit jejich parametry na základě predikce povrchových vlastností může potenciálně zlepšit bezpečnost, komfort a náklady na údržbu moderních automobilů.

Jsou použity dvě veličiny popisující vlastnosti povrchu: hrubost a tření (kluzkost). Dataset je získán automatickým způsobem, není potřeba manuální anotace. Při tréninku se získávají hodnoty veličin automaticky podle vlastností odvozených z reakcí vozidla (vertikální zrychlení, podélný skluz, točivý moment atd.). Při nasazení systému se používá k predikci vlastností povrchu pouze obraz.

Tato práce navazuje na výsledky dosažené v naší předchozí práci (Čech a kol. 2021[1] a Vošahlík a kol. 2021 [2]). Předchozí modely byly omezeny na předpovídání jediné hodnoty vlastnosti povrchu pro celý obrázek. Jsou navrženy dva přístupy. První využívá k predikci mapy povrchových vlastností konvoluční neuronovou síť U-Net natrénovanou na polosyntetické datové sadě. Druhá metoda využívá vizualizace z hlubokých sítí prostřednictvím lokalizace založené na gradientu (Grad-CAM). Všechny testy byly provedeny na experimentální platformě. Oba modely byly ověřeny na ručně anotované testovací datové sadě a ukazují slibné výsledky.

Klíčová slova: vlastnosti povrchu, konvoluční neuronové sítě, automatická anotace

Překlad názvu: Predikce vlastností povrchu z obrazu s využitím samoučení

Contents

Figures

Tables

Chapter 1

Introduction

Performing any maneuver in a vehicle is hugely influenced by the surface properties. The forces generated in the wheel to road interface are deciding factors of safe driving. The traction properties of different surfaces are significantly influenced by their materials. That has an impact on the vehicle braking distance and overall controllability and stability of the car.

Professional drivers are able to predict the road situation ahead, taking into account tire and surface conditions and adjust their driving style accordingly. Most of the everyday drivers do not have such an experience, thus many of the advanced driver-assistance systems (ADAS) are being introduced into modern cars. Even though ADAS systems such as adaptive cruise control (ACC), anti-lock braking system (ABS), electronic stability control (ESC), traction control system (TCS) and many others help human drivers significantly with their driving, most of these systems function reactively. That means that they start intercepting into driving only based on the measurements and traction system response when the car is already present on an incriminated surface.

Predictive visual recognition of the surface properties in front of a vehicle has potential to improve safety, comfort and even costs coming with maintenance. Deep learning techniques in image recognition have become more useful and accurate in the last few years. The critical part of image recognition system developement is its training. The training phase relies on a dataset size and its labels accuracies. Manual data annotation costs significant amounts of money and can be extremely tedious. The surface property annotation is even more complicated since it is a rather continuous quantity and is unclear how it should be labeled.

Self-supervised method of surface visual prediction is a promising solution in our case. We can collect large amount of data that are automatically labeled using the sensors mounted on a vehicle. Such an attitude even allows for continuous dataset extension and improvement.

The goal of this thesis is to implement such data harvesting on our subscale vehicle platform. The surface properties prediction will then be trained

using this data. The objective is to present not only prediction of a single regression value for each image but to perform pixel-wise segmentation of the surface properties. This will be done using different approaches such as synthetic dataset creation or using visual explanations from deep network via gradient-based localization. The second approach builds on top of our previous work from papers [1] and [2]. The main contribution of this thesis are:

- Robust birds-eye view of an area of specified dimensions
- The creation of a semi-synthetic dataset
- U-Net method for prediction of a local surface properties map
- Method using visual explanations from deep networks via gradient-based localization (Grad-CAM) to obtain a local surface properties map.

The rest of this thesis is constructed as follows: first are researched existing techniques and related literature. The next step will be a robust orthorectification of an original image into an area of given dimensions from a birds-eye view. We will describe dataset collection and semi-synthetic images creation. Finally, two methods using convolutional neural networks will be trained to output local surface properties maps.

Chapter 2

Related work

There have been several image-based approaches of mapping the surface in front of a vehicle. Some of the methods are concerned with detecting damages and anomalies such as potholes or cracks in the road. The work of Minh-Tu Cao [15] has surveyed the performance of several deep learning models such as convolutional neural network and Faster Region-based convolutional neural networks (Faster R-CNN). Lydon [16] has reviewed different methods of surface defects, deformation and cracking. He compared customized vehicle setups using laser scanners, small-scale vision-based monitoring setups, stereo vision and Resnet neural network with the convolutional neural net ending at the top.

Other works are concentrated on surface classification rather than defining it as a continuous regression quantity. Team from Ghent university [17] developed a method of unsupervised content-based road type classification using image data. The paper [3] uses surface detection and image processing to recommend route.

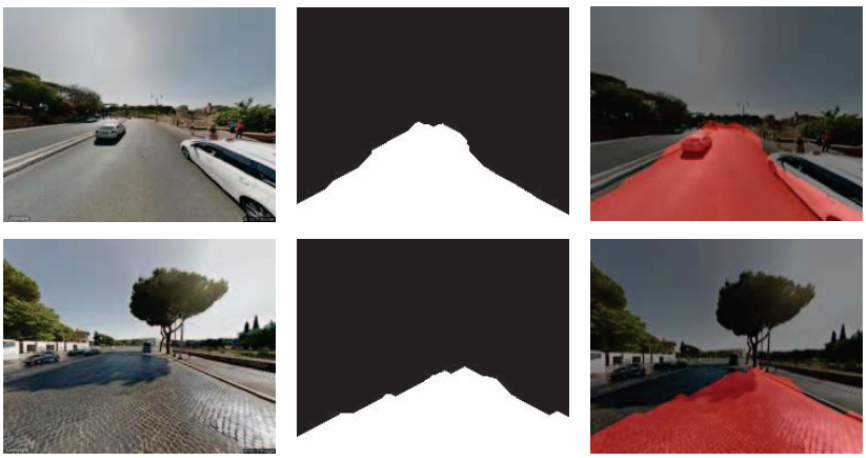


Figure 2.1: Road surface area extraction results. Original image in the first column. Cutting results after the final refinement in the second column. The image marked with a road area covered in third. Adopted from [3]

The paper [18] uses a binary classifier to detect paved and non-paved road classes. It is also done using ResNet50 implementation. Same method is used in [19]. There have also been published methods of identification of surface type for friction coefficient estimation [20]. One of the methods has been published by our predecesors in Tomi project [4]. The surface type classification and related road friction properties are provided to the ABS (braking control algorithm) in order to adjust the vehicle response accordingly.

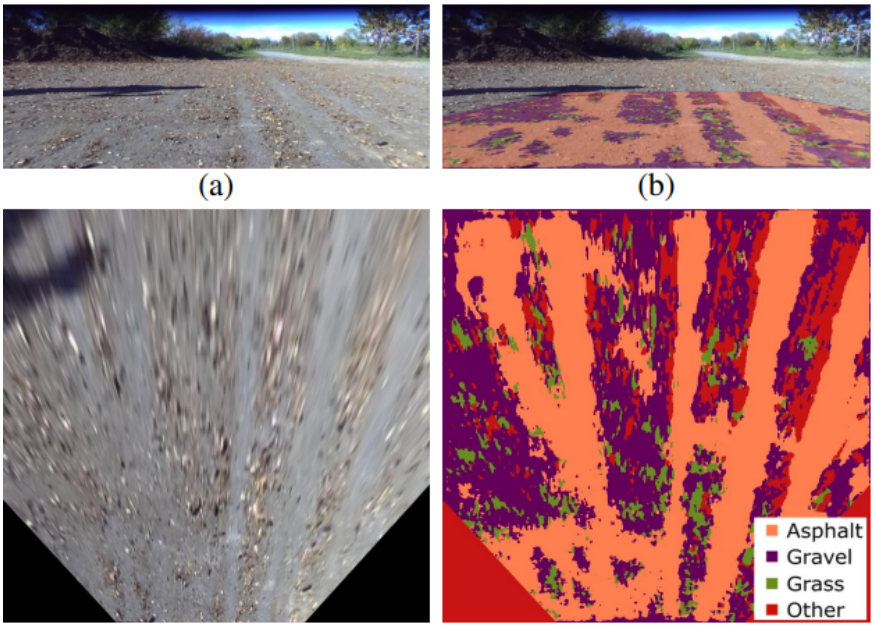


Figure 2.2: Visual surface recognition - Tomi1 team [4]

The methods mentioned above rely heavily on manually labeled data. That is a costly and exhaustive process. That is the reason we choose the automatic annotation, which allows us to process huge amount of data and create sufficient datasets expected by deep learning methods.

The principle of having two correlated signals and using one as a label for classification or regression to predict the other is well known. The cross-modal training is used in many different areas. Facebook AI Research [5] uses a single camera to percept depth, estimate the normal vector of a surface and semantic labeling using ground truth depth maps. The results are visible in Fig. 2.3.

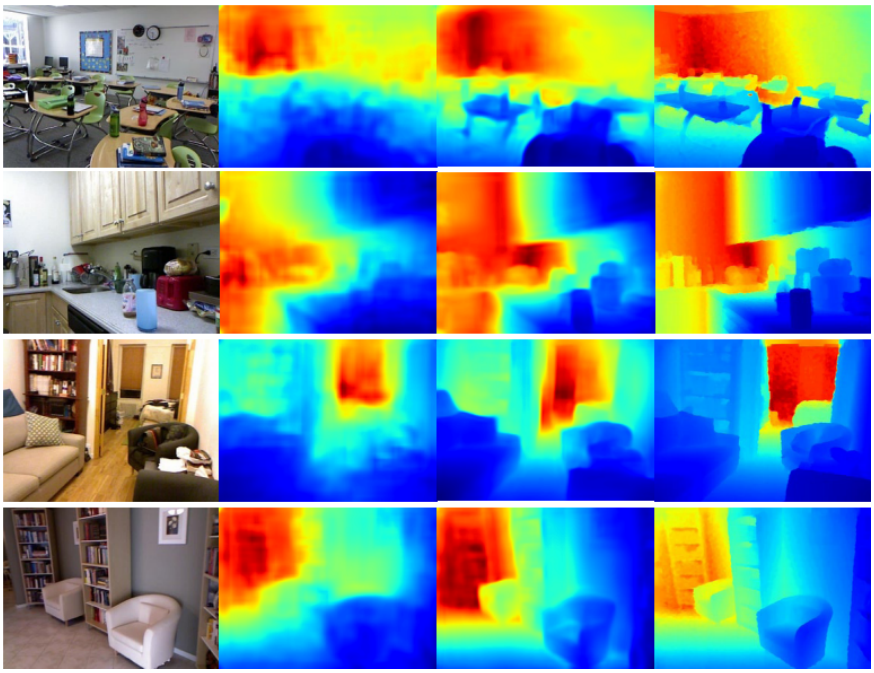


Figure 2.3: Example depth results from [5]

A similar approach is nowadays being heavily used in the medical field. The advances in image processing help doctors to detect potential problems in the early stages. Convolutional neural networks are used to detect brain tumors [21] helping radiologists make better and quick decisions.

In the automotive industry, there exist methods combining data measured by accelerometer and images harvested by a camera mounted on a vehicle. The paper [22] uses accelerometer data to detect damages and defects on asphalt as a countermeasure to car accidents. The same problem is solved in [23] using smartphone camera and accelerometer. Though these methods deal with a similar problem to ours, they do not use the accelerometer data to train the full surface segmentation of the surface.

Finding the traversability of the terrain has been unfolded in [24]. For the model training were used the measurements from the range sensing lidar. The obstacles found by the lidar are projected into the occupancy grid, which is the used as a label for the camera images. The process is then unsupervised, deriving the traversability property using just the geometry and appearace of the scene. This unsupervised concept is closest to our goal, only using different modality as a label.

Mercedes Benz has developed a system called Magic Body Control. They use the stereo camera to scan the road ahead for bumps and dips. Using the received information the suspension system is then adjusted (Fig. 2.4). It works up to about 15 meters to ensure maximal smoothness for the passengers.

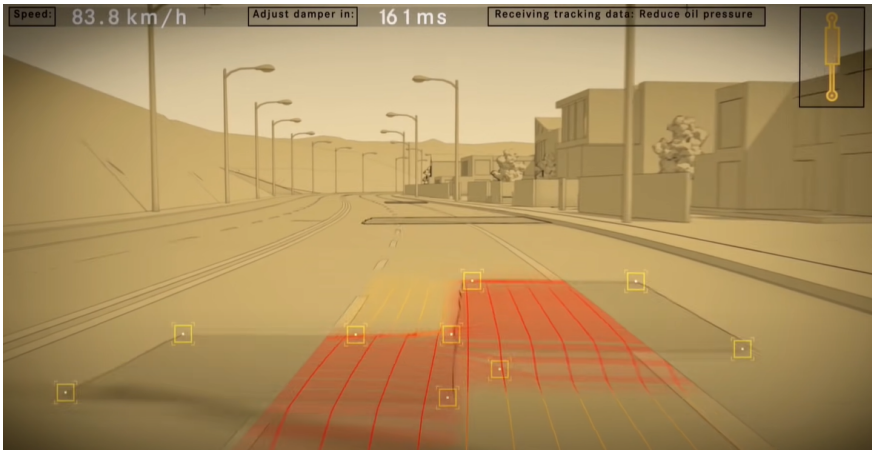


Figure 2.4: Surface scan of Mercedes Magic Body Control [6]

Audi has introduced a similar approach firts in their top-end A8 model. The system is called Predictive active suspension. It can load and unload each wheel separately to attain the best chassis position to ensure maximal comfort. This system also works using front looking camera detecting anomalies on the surface ahead and predictively regulating the actuators (Fig.2.5).

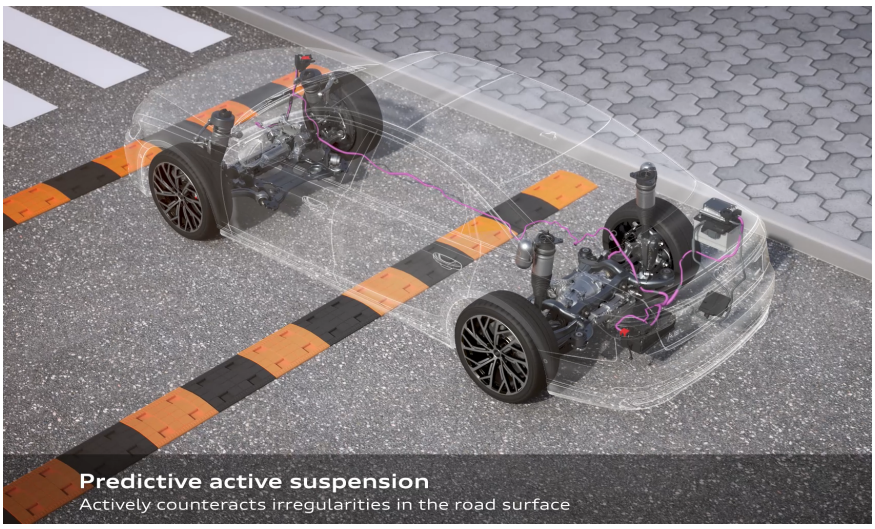


Figure 2.5: Audi Predictive active suspension [7]

It is clear that predictive techniques for ADAS systems are used and wanted by the car manufacturers. Though these corporations do not share the information about the datasets, training methods or models architectures it is obvious that the computer vision techniques are implemented into those systems. It seems that there is a future for the surface properties prediction for the automotive industry.

Part I

Theoretical background

Chapter 3

Surface properties

This section serves as a description of two parameters used to characterize surface in front of a vehicle. These two concepts have been used in papers [1] and [2]. The abstractions of surface properties representations are described in sections 3.1 and 3.2.

3.1 Surface roughness

The roughness of the surface can be measured in many ways. These methods are comprehensively described in [25]. The accelerometer-based techniques implemented specifically to detect and characterize road anomalies are described in [26]. The use of vehicle acceleration measurements to estimate road roughness [27] is the closest implementation of measuring surface roughness to ours. We are using the attitude of analyzing the measurement signal in the time domain rather than in frequency one. A similar approach has been used DARPA Grand challenge by Stanley [28]. They used the measured magnitude of the accelerometer signal to adjust the driving speed of the autonomous vehicle.

In our implementation, the surface roughness is derived from the vibrations signal measured by the accelerometer mounted on the front axle next to the left wheel (Sec. 5.1). Having $a_z(t)$ as an accelerometer signal, we define the surface roughness as:

$$\rho(t_0) = \frac{k}{v(t_0)} \sum_t W(t - t_0) a_z(t)^2. \tag{3.1}$$

This definition is adopted from [1]. From the equation: $a_z(t)$ is the vertical acceleration (without the gravity) over time, $W(t-t_0)$ is the Gaussian window to filter the noisy accelerometer signal, $v(t_0)$ is the vehicle velocity at time t_0 and k is the constant ensuring the roughness is from the range of 0 to 1.

Normalization by the velocity is important since the magnitude of the acceleration signal is linearly dependent on the velocity itself. The linear de-

pendency has been measured experimently and is in agreement with [28]. We have performed a vast majority of experiments measuring surface roughness at a predifined constant speed using automatic cruise control. We believe that it results into most accurate measurements and normalized surface roughness labels for our dataset.

3.2 Surface friction

The road friction condition is being measured in many different areas. Since it has a huge impact on the maneuver safety and road-tire interface it has found its place not only in the automotive industry but also in aerospace. The large scale measurements are continuously performed on airport runways by grip testers [29].

One of these machines operates every day at the Prague Václav Havel Airport. In Fig. 3.1 and Fig. 3.2 is a visible mechanism of the fifth wheel capable of accurately measuring surface roughness. The adhesion of the airport runway is crucial for the safe landing (and take offs) of aircraft.

Figure 3.1: Trunk of the car measuring surface friction on Prague airport [8]

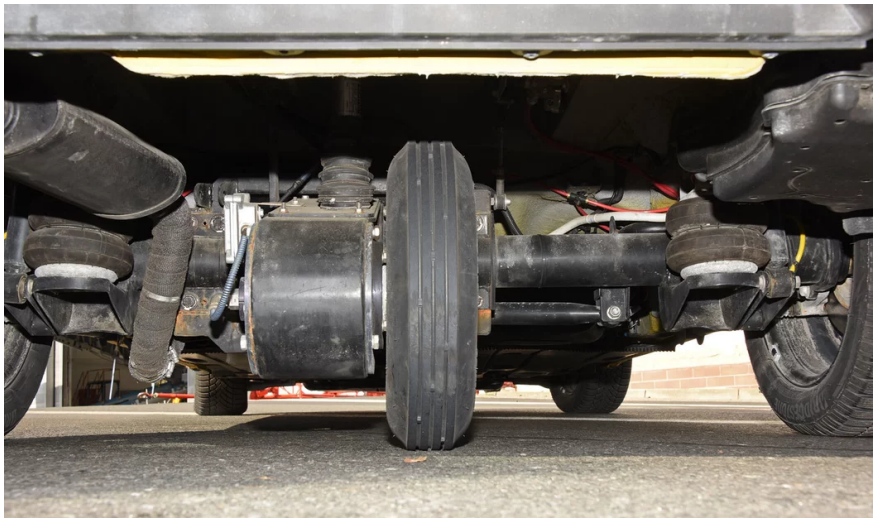


Figure 3.2: Car measuring surface friction on Prague airport from below [8]

Our implementation of the surface friction estimation is a part of a Ph.D. studies of Ing. David Vošahlík. His algorithm provides a measurable value of a surface friction ϕ, which is used as a label for the driven surface [2]. He has developed an Unscented Kalman filter to estimate the friction.

The tire-to-road interface is modeled by the well-known Pacejka magic formula [30]. The high fidelity twin-track nonlinear model is used as a simulation model for the estimator. The vehicle dynamics is modeled with suspension and each wheel by Pacejka magic formula Eq. 3.2 and Kamm's circle friction ellipse.

$$F_x(\lambda) = \mu F_z D \sin(C \arctan(B\lambda - E(B\lambda - \arctan(B\lambda)))) \tag{3.2}$$

The slip ratio λ is used as the slip variable, μ is the road friction coefficient, B, C, D and E are Pacejka coefficients characterizing the slip curve shape, and F_z stands for wheel normal force [2]. The surface friction is defined as part of the magic formula:

$$\phi = \mu F_z D. \tag{3.3}$$

The whole algorithm relies on the following assumptions: the wheel angular velocity is measured, the wheel pivot point velocity vector is measured or estimated, parameters of the wheel dynamics are identified and that all resistant torques such as rolling resistance induced torque, friction torque, and others are known.

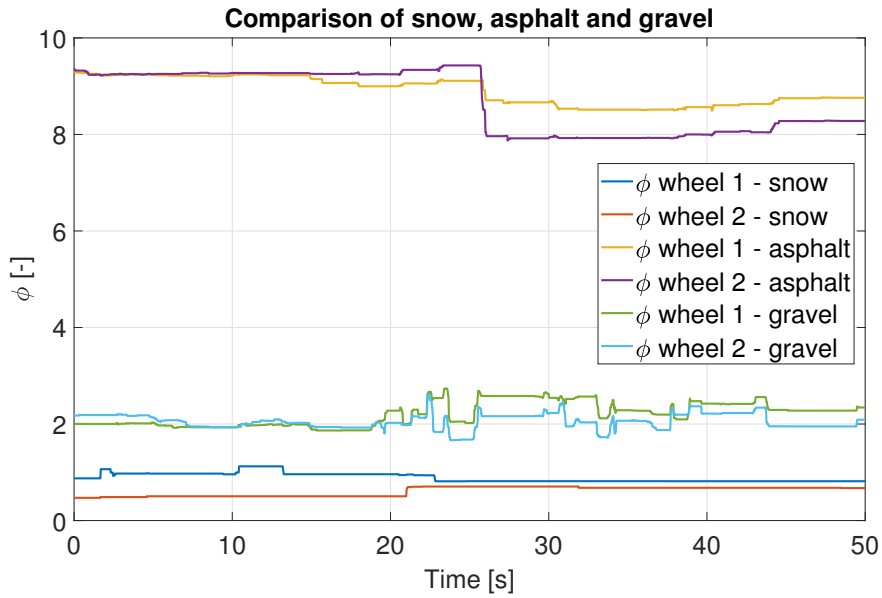


Figure 3.3: Comparison of the estimated friction for different surfaces based on the subscale platform measured data [2]

In Fig. 3.3 are visible estimated friction coefficients from the data measured from rides on our Tomi2 platform (Sec. II). More thorough explanation of the whole surface friction estimation algorithm pipeline is in the article [2].

Chapter 4

Machine learning

4.1 Supervised, unsupervised, and self-supervised learning

There are many terms used in machine learning to describe different concepts in model learning based on the availability of labeled data. This section will describe the basic regimes and explain which are used in this thesis. The definitions are adopted from [31].

4.1.1 Supervised learning

In supervised learning, the training multi-set of examples is available [31]. The labels (annotations) are known for all observations in the dataset. Classification (Sec.4.1.5) and regression (Sec.4.1.6) are typical examples of supervised learning. The popular approach these days would be to train a deep neural network on an available dataset with the loss being the difference between the predicted output and the actual true label. Supervised learning can be considered as the most straightforward kind of a machine learning method.

4.1.2 Semi-supervised learning

Unlike supervised learning (Sec.4.1.1), semi-supervised learning assumes to have the true labels only for some partition of the available dataset. This problem of semi-supervised learning is often used by pseudo-labeling. Pseudo-labeling starts with training a model in a fully supervised way on the part of the dataset that allows such thing. This model is then used to artificially label the rest of the data. This approach does not produce a perfectly annotated dataset, so only the labels with high confidence from the model are added to the final dataset.

4.1.3 Unsupervised learning

Unsupervised learning is used when the training dataset is available but without the corresponding labels. These labels must be found in the data itself by some sort of data analysis [31]. We usually seek to find some underlying patterns in the data. Data clustering is usually applied to achieve that. There are multiple clustering algorithms such as K-means [32].

Unsupervised learning is widely used in online marketing [33]. It is used for customer segmentation. With the understanding of different customer groups has the company great advantage in building specific marketing strategies for each cluster. Unsupervised learning is also being used for fraud detection by betting companies. They possess huge datasets of bets. Models trained on this data can easily detect anomalies in betting and match results.

4.1.4 Self-supervised learning

Self-supervised learning could be seen as an autonomous form of supervised learning. In our case, it requires no human input and all the data are automatically annotated by the system itself. This approach is weel scalable, since no human intervention is not needed.

4.1.5 Classification

The goal of the classification task is to determine one of the predefined output class for the input vector. The input can be an unseen image (three-dimensional matrix of width, height, and three RGB channels) resulting into a predefined output class. Model trained on the ImageNet dataset [34], which contains 1000 classes. We then want the model to predict as many true positives as possible. There is actually competition for such models established in 2005 called ImageNet challenge. The goal is to achieve the best error rate on the given dataset.

4.1.6 Regression

Regression differs from the classification in a fact, that it predicts a continuous numerical value, rather than one of the predefined discrete class labels. An example is a convolutional neural network model that predicts human age based on their image [35]. Regression will be used for our models in this thesis since we want to describe surface by a continuous property.

4.2 Neural networks

A neural network is one of the machine learning methods. Modern computer vision methods heavily use artificial neural networks. This thesis works with convolutional neural networks (Sec.4.4), which are one type of artificial neural nets. These methods are experiencing great boom in the last decade with the improvements of computational powers of modern CPUs and GPUs.

4.3 Backpropagation

Backpropagation is one of the core techniques of neural networks. It is the method of learning for the neural net. It is basicaly a recursive application of the chain rule along the computational graph of the neural network for the gradients computation of a loss function with respect to the weights. The implementation maintains a graph structure.

First, the forward pass is computed, and the result of the operation is saved. Any intermediate results needed for the gradient computation are also stored in the memory. In the backward pass is applied the chain rule to compute the gradient of the loss function with respect to the inputs (Fig.4.1). The model weights are then updated and the process repeats itself. The loss function is being optimized.

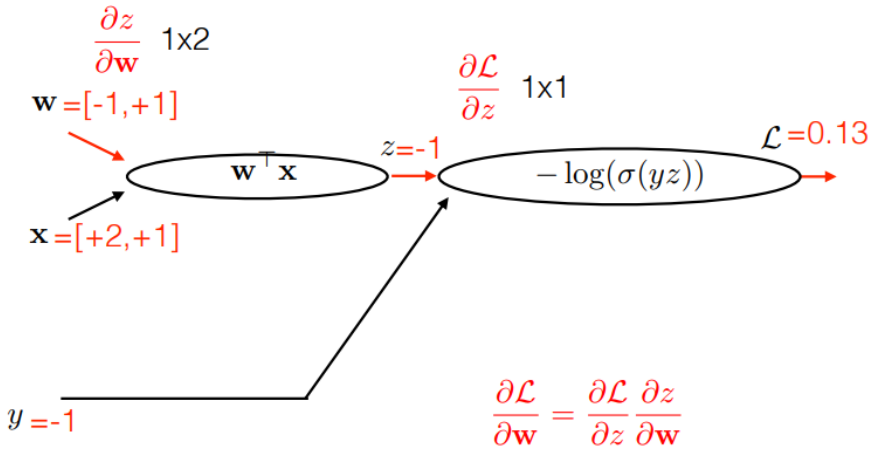


Figure 4.1: Backpropagation in vector representation [9]

4.4 Convolutional neural networks

Convolutional neural networks have become frequently used in computer vision tasks. They contain at least one convolutional layer and usually

pooling blocks. They can adaptively learn spatial feature hierarchy through backpropagation. They are being used in medical field (radiology [36], [21]) and also automotive [6].

4.4.1 Convolutional layer

A convolutional layer is the main building block of the convolutional neural networks. This layer works by taking the filter and sliding it over the image spatially and computing dot products at every spatial location (Fig.4.2). It basically centers our filter on top of every pixel in its input volume, starting at the upper-left corner, and at every position it is going to compute this dot product (that produces one value).

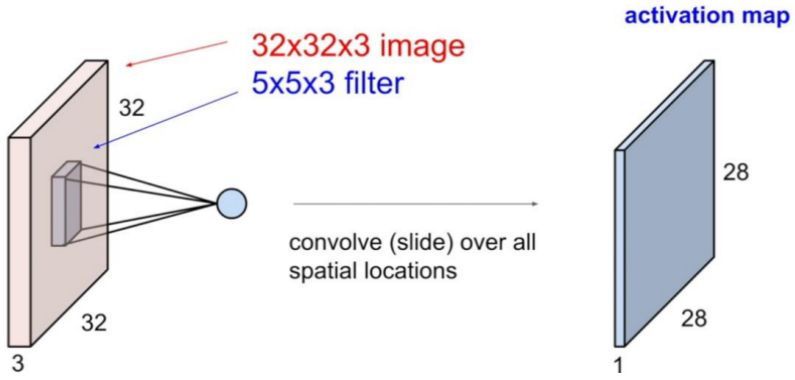


Figure 4.2: Example of the convolutional layer [10]

Each convolutional layer can have multiple filters, each producing one activation map. The filters at the earlier levels (close to the image input) tend to represent low-level features such as edges. When we get further in the model we start getting more complex kind of features and the final layers usually start to resemble the classes.

There are several parameters set in the convolutional layer:

- Stride - the shift of the filter in the input. Example : Fig.4.4.
- Padding - controls the amount of padding applied to the input. Example: Fig.4.3.
- Dilation - controls the spacing between the kernel points.
- Kernel size.

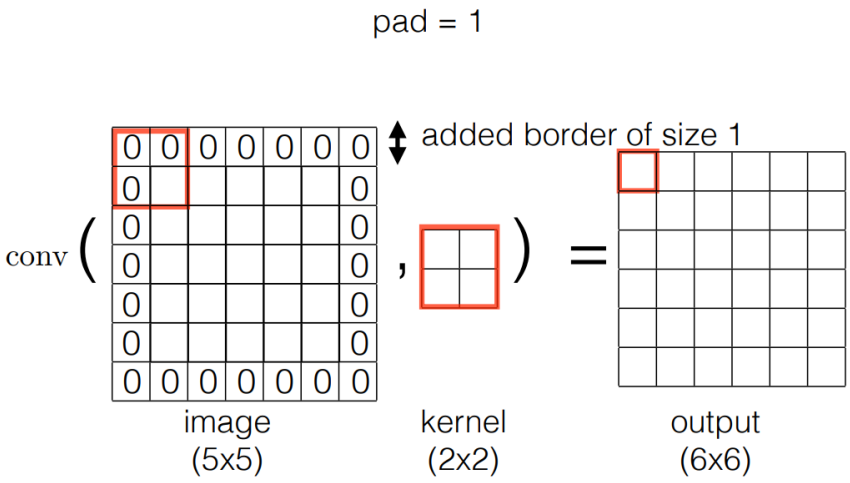


Figure 4.3: Zero padding of 1. [9]

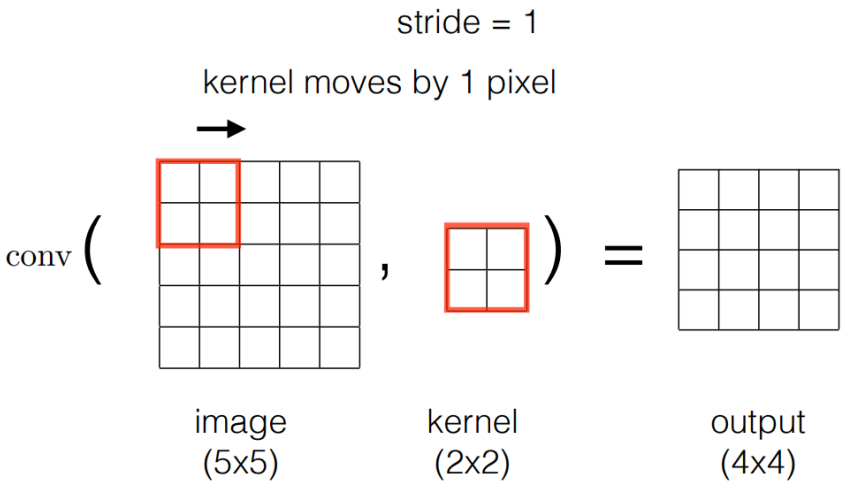


Figure 4.4: Example of a stride [9]

The dimensions (MxM) of the output activation map produced by the convolutional layer can be computed as:

$$M = \frac{N + 2P - K}{S} + 1, \tag{4.1}$$

where N is the input width, P is padding, K is the filter (kernel) size and S is stride.

4.4.2 Pooling layers

Pooling layers are mainly used to make the representations of inputs smaller and more managable. It operates over each activation map independently. The pooling layer replaces a certain location by some statistic. There are several different examples of pooling layers:

- Max pooling - takes the maximal value of some region: Fig.4.5
- Average pooling - takes the average value of some region
- Global pooling - downsampling taken to the extreme, entire feature map downsampled to a single value

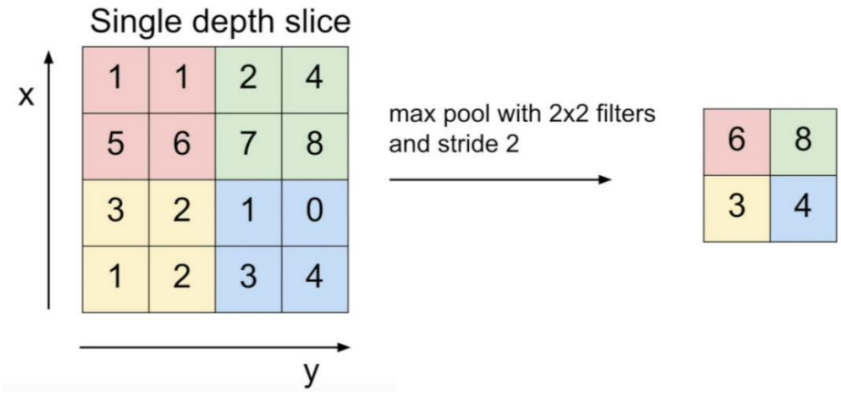


Figure 4.5: Example of a maxpool layer [10]

4.4.3 Activation functions

Activation fuctions are functions applied to neuron inputs. In the convolutional neural networks usually used right after convolutional layers. The activation functions basically decide wether a neuron should be activated or not. Activation functions decides the importance of an intput for the network prediction. There are several activation functions:

- Sigmoid
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$
- Tanh
$$y(x) = \tanh(x) \tag{4.3}$$
- Relu
$$y(x) = \max(0, x) \tag{4.4}$$
- Leaky RELU
$$y(x) = \max(0.1x, x) \tag{4.5}$$

4.4.4 Loss functions

The loss function is a function to define the success of the model prediction. We want to minimize such function. Arguments of this function are the real value and the model output. There are several loss functions, here is an example of some:

- Mean square error ($\mathcal{L}_2$ loss)

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n} \tag{4.6}$$

- Mean absolute error ($\mathcal{L}_1$ loss)

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{4.7}$$

- Cross Entropy Loss (used for classification)

$$CEL = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)) \tag{4.8}$$

4.4.5 Batch normalization

To reduce the strong dependence of the model regularization we use batch normalization. It is usually used after fully connected or convolutional layers. The batch normalization improves gradient flow through the network and allows for higher learning rates. The strategy is to compute mean and variance independently for each dimension and normalize the given batch.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \frac{1}{n}\sum_{i=1}^{n} x_i}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \frac{1}{n}\sum_{i=1}^{n} x_i)}} \tag{4.9}$$

4.4.6 Regularization

Regularization is a concept of penalizing the complexity of a model rather than explicitly trying to fit the training data. There are also many techniques of regularization. $\mathcal{L}_2$ regularization and $\mathcal{L}_1$ regularization penalize the weight vector using the hyperparameter that controls the importance of the regularization. Another regularization technique is called dropout. It randomly ignores some amount of neurons during the training phase. The aim is to prevent overfitting on a provided dataset and improve the model to work on an unseen data.

4.5 Used architectures

The following architectures and methods have been implemented in this thesis.

4.5.1 ResNet

Deep neural networks are difficult to train. In the article Deep Residual Learning for Image Recognition [11] the autors introduced a skip-connection learning block (Fig.4.6). The building block contains a shortcut over some layers. There is typicaly activation function and a batch normalization in between. This approach is proven to help with a problem of vanishing gradient during backpropagation. This allowed the authors to train a network with 152 layers that performed better than human on ImageNet [34]. Resnet50 architecture is shown in Fig. 4.7.

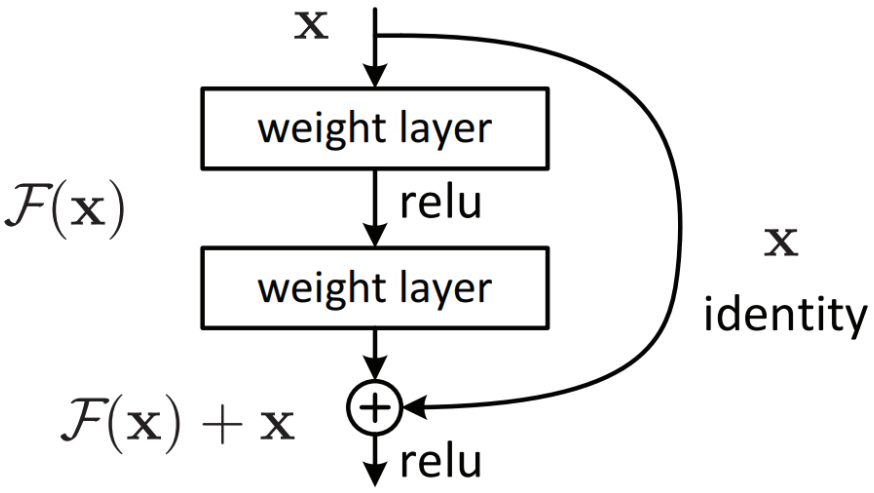


Figure 4.6: Resnet building block [11]

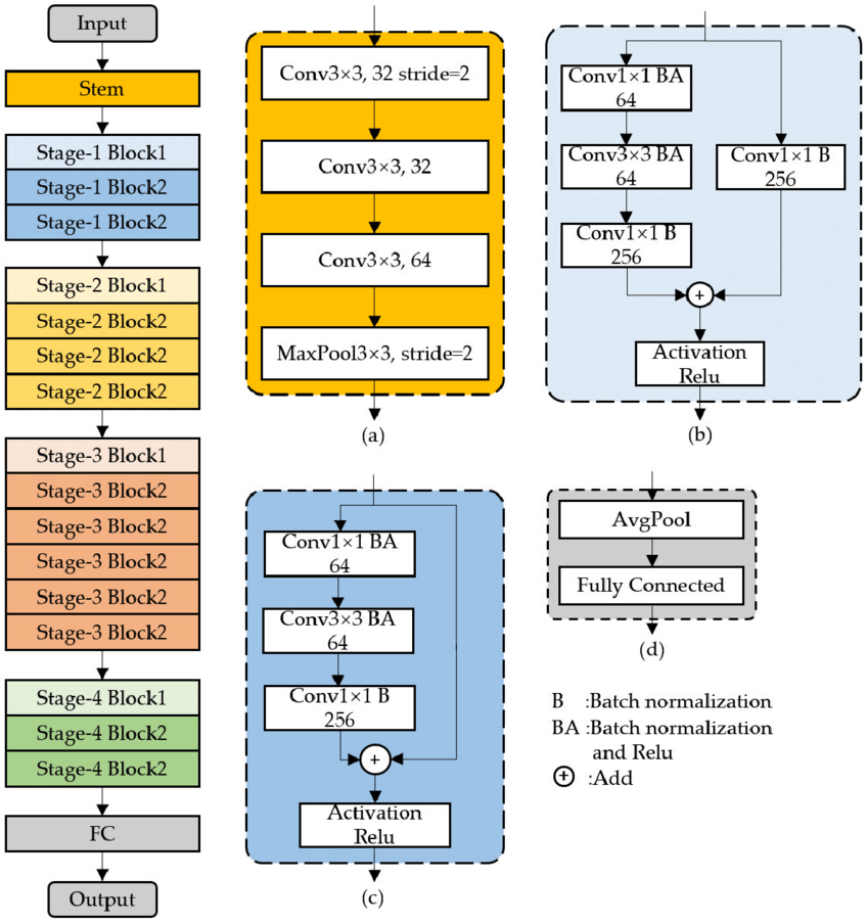


Figure 4.7: Resnet architecture [12]

4.5.2 UNet

Unet architecture has been first found in Convolutional Networks for Biomedical Image Segmentation paper [13]. It is architecture fitted to the image pixel-wise segmentation. The architecture consists of tho main path 4.8. First is a contracting path followed by a symmetric expanding path. The contracting path (encoder) captures the context of the input image. The spatial information is reduced but the number of channels increases. The expanding path (decoder) enables precise localization of the features found.

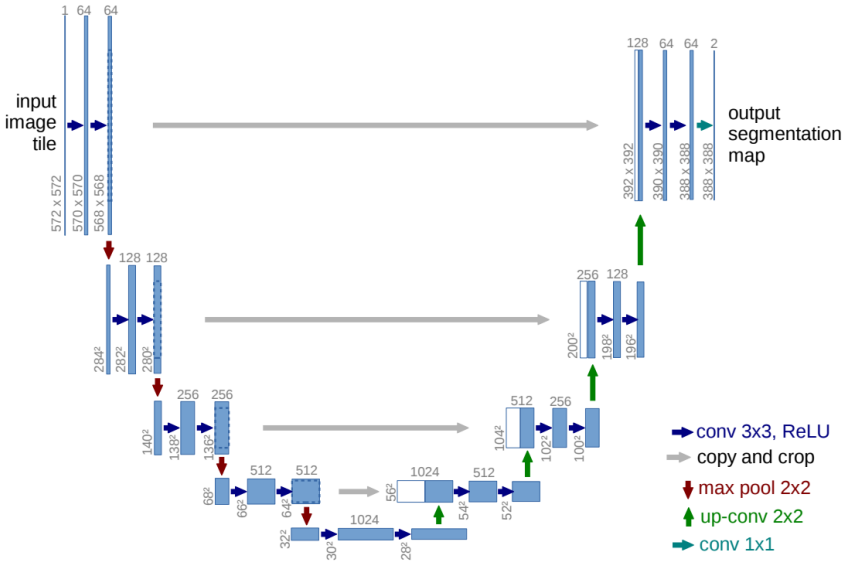


Figure 4.8: Unet architecture [13]

4.6 Grad-CAM

Modern deep learning models have become very complex and accurate. But they can sometimes act as a blackbox. That forced the development of computer vision analysis tools. The Grad-Cam algorithm comes from the Visual Explanations from Deep Networks via Gradient-based Localization [37]. The paper proposes a technique for producing visual explanations for decisions from large convolutional neural networks models. This helps them to be more tranparent and explainable.

The intuition is based upon the fact that the model must have decided the output based on some regions of the image. The Grad-CAM is used for classification networks. It starts with finding of the gradient of the most probable class with respect to the latest activation map in the model.

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i\sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \tag{4.10}$$

The y^c is the score of the class, A^k is the feature map activation of a convolutional layer.

To obtain the algorithm output, a weighted combination of forward activation maps is performed, followed by the ReLU.

$$L^c_{\text{Grad-CAM}} = \text{ReLU}(\sum_k \alpha^c_k A^k) \tag{4.11}$$

The coarse heatmaps are projected into the original image in fig. 4.9.

(a) : Grad-CAM Cat [37] **(b) :** Grad-CAM Dog [37]

Figure 4.9: Visual explanations by Grad-Cam

4.7 RANSAC

Random sample consensus (RANSAC) [38] is an algorithm for a robust estimation of a model from measured data. RANSAC is especially useful for the data containing outliers that would otherwise influence final model of different algorithm. RANSAC is suited for applications in image analysis and pointcloud filtering [38]. It is part of the non-determenistic algorithms family. The result of RANSAC is influenced by the number of iterations, minimum number of samples chosen randomly from the original data and chosen threshold. Below follows the pseudocode of the random sample

consensus algorithm 1 with all the necessary parameters for its function.

Data:

- pointcloudData - pointcloud points
- n – minimum number of data points required to estimate model parameters
- t - threshold
- stopInliers - stop iteration if at least this number of inliers are found
- k - maximum number of iterations

Result: bestModel - model which bests fit given data (parameters of a plane)

```
iterations = 0;
bestModel = null;
bestError = ∞;
while iterations < k do
    ranInliers := n randomly selected data points from
     pointcloudData;
    Model := model fitted to ranInliers;
    Inliers := ∅;
    for each point p in data and not in ranInliers do
        if point fits to Model with error < t then
            add point p to Inliers
        end
    end
    if # of elements in Inliers > stopInliers then
        newModel := model fitted to ranInliers and Inliers;
        error := error of newModel for ranInliers and Inliers;
        if error < bestError then
            bestModel := newModel;
            bestError := error
        end
    end
end
```

Algorithm 1: RANSAC pseudocode [39]

RANSAC basically loops over two main steps (picking random samples and verifying model fitted to them). It does not need huge amounts of memory, thanks to random sampling of the given data. [40] Rather than using whole dataset and attemping to eliminate outliers, it uses small initial data set and enlarges this small set with eligible data when possible. [38]

Part II

Hardware platform

Chapter 5

Tomi2 overview

The Tomi2 platform is based on a commercial RC platform. It has BLDC electrical, motor and it is in 1:5 scale. Our particular model is Losi1:5 DBXL-E. It has 844x501x308 mm dimensions. The weight of the bought platform is 12.5 kg rising to 20 kg, taking into account all of the additions made. Each wheel has its own servo motor for independent four-wheel steering. In the car are mounted two 14.8 volts LiPo batteries.

Several computing units have been added to the platform. Nvidia Jetson Xavier is the most relevant computer to this work. It has a powerful GPU compatible with Nvidia CUDA system, so it supports running neural network models implemented in Pytorch. Nvidia Jetson has a rather friendly input voltage range of 9-20 volts. This range perfectly accepts additional 14.8V LiPo battery. The Nvidia Jetson can demand up to 72 watts of power in peak. The LiPo traction battery can provide several amperes to the motor controller, so using it to power Nvidia Jetson leaves us with plenty of headroom.

Raspberry Pi4 can be considered as a main hub that connects all the modules placed on this platform. It is connected to the Nvidia Jetson Xavier via ethernet. It also communicates with another Raspberry board with Navio shield. Its main function is to communicate with the motor controller and send PWM signals. Navio also receives signals from radio controller. Navio has its own IMU and gyroscope readings and Ublox GPS.

Another module is STM Nucleo. It processes signals from Hall sensors placed on each Tomi2 wheel. The information about RPMs is the sent via UART to the main Raspberry. There is also Arduino Nano that processes the signal from the accelerometer (Sec.5.1) and sends it to Raspberry via UART.

The Tomi2 platform is developed in collaboration with Toyota Research Lab under supervision of doc.Ing. Tomáš Haniš, Ph.D. and Ing. Jan Čech, Ph.D. Team members are Tomáš Twardzik, Marek Boháč, Jan Švancar, David Vošahlík, Tomáš Rutrle and myself.

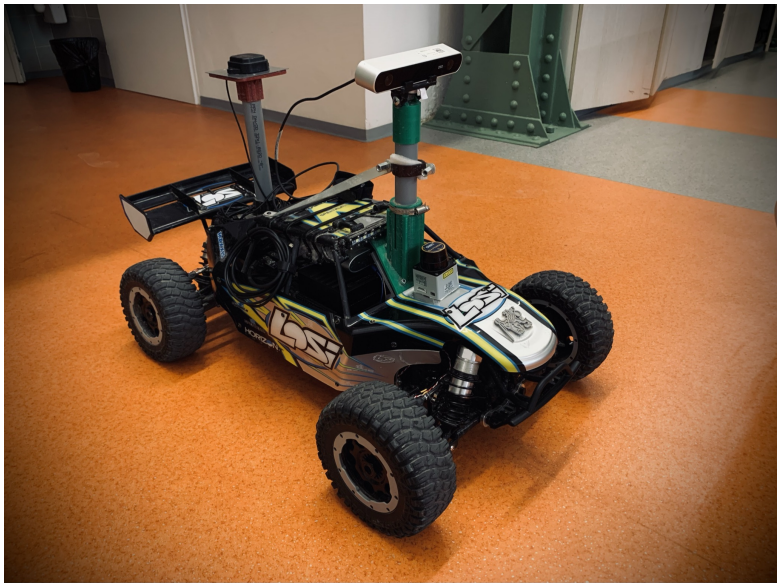


Figure 5.1: Tomi2 platform

Figure 5.2: Tomi2 platform at university yard

5.1 Accelerometer on the front axle

The main sensor that is used to measure surface roughness (Sec.3.1) is located on the front axle next to the left wheel (Fig.5.3). It is an analog accelerometer from Analog Devices with the model name ADXL326. It is compact, complete three-axis accelerometer. It measures acceleration with a minimum full-scale range of $\pm 16g$. Users can define bandwidth of the accelerometer starting at 0.5 Hz and going up to 1600 Hz. It has a small profile with dimensions of 4 by 4 by 1.45 mm. You can see the 3D-printed housing by Tomáš Rutrle in the Fig. 5.3.

Figure 5.3: Accelerometer location [1]

5.2 ROS2 nodes

The Tomi2 platform runs ROS2 system used for communication between processing units on the vehicle. The ROS nodes are:

- ZED2 wrapper - Publishing on the camera image and pointcloud topics
- Image processing node - Binary classifier for tiled pavement

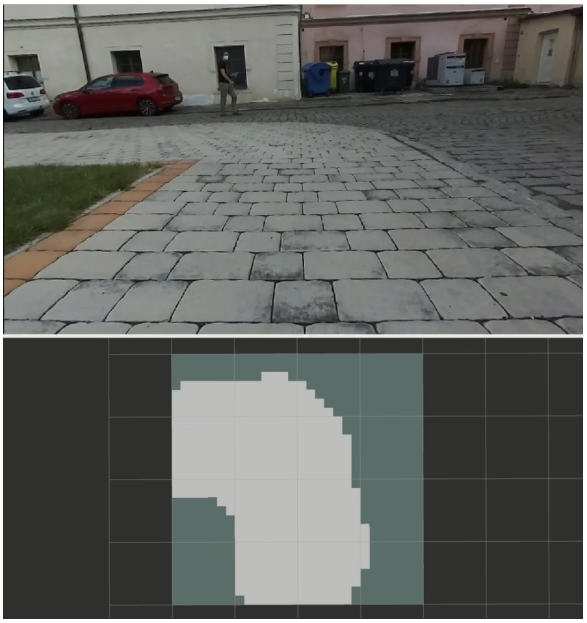


Figure 5.4: Binary classifier in image processing node

- Motion planning - Finds path in the map received from Image processing node using RRT

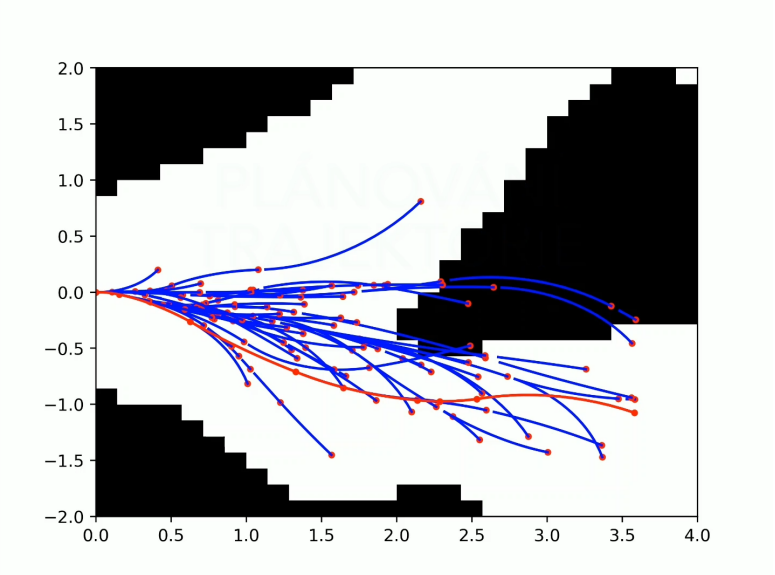


Figure 5.5: Path planning

- Trajectory tracking - Subscribes found path and publishes velocity and steer commands
- Controls - The main control loop, options are manual control, cruise control, slip ratio control and autonomous driving.

5.3 Data acquisition

Data recorded during rides:

- Rostime
- Navio IMU - (accelerometer, gyroscope, magnetometer)
- Longitude, Latitude
- Wheel RPMs - Hall sensors
- Heading and speed from GPS
- Signals from 10-channel receiver
- Accelerometer data from the front axle
- ZED2 camera images.

5.3.1 ZED2 camera

We are using a ZED2 camera from Stereolabs, which you can see in Fig. 5.6. This camera has 120° wide-angle field of view. Since it is a passive stereo camera it has two lenses, both with 16:9 native ratio. ZED2 has a built-in IMU sensor with additions of barometer and magnetometer. The camera itself with the addition of its SDK provides multiple advanced systems, such as skeleton tracking, simultaneous localization and mapping, remote monitoring and, data collection and spatial object detection.

Figure 5.6: ZED2 [14]

The most useful feature for our project has been the depth-sensing ability of ZED2 camera. The fact that this camera has two lenses allows it to estimate depth and motion by comparing the displacement of pixels between the right and left images. It provides a depth map consisting of (x, y, z) distance values in relationship to the position of the left lense. Stereolabs labs SDK allows the pointcloud to contain colour information as well as space coordinates, which you can see in the Fig.5.7

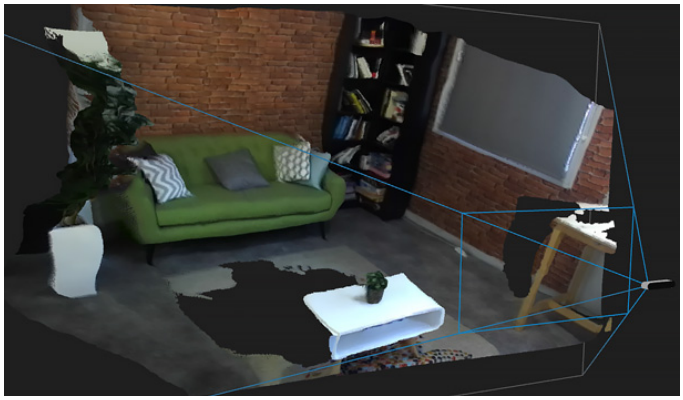


Figure 5.7: ZED2 pointcloud fused with image [14]

ZED software developement kit provides several settings of the depth map that can be tweaked to our liking. ZED2 camera has two main modes of depth sensing: standard and fill. Standard mode runs faster than the fill mode, but it contains holes due to the computational algortithm. This mode is used for autonomous navigation and obstacle detection, where speed and reliability are the main concerns. We can also set the depth range in front of the camera that is taken into account. It can provide a depth map from 0.3 to 40 meters.

5.3.2 Camera recording

ZED2 allows us to record in many image formats such as HEVC, JPG and PNG, meaning that it can do lossless compression. SDK can be used to store the video alongside with timestamp or IMU data. There are many options for the camera output as you can see in the Tab. 5.1. We can adjust many camera settings such as brightness, contrast, hue, saturation and many others.

Video mode	FPS	Output resolution (side by side)
2.2K	15	4416x1242
1080p	30 / 15	3840x1080
720p	60 / 30 / 15	2560x720
WVGA	100 / 60 / 30 / 15	1344x376

Table 5.1: ZED2 camera resolution [14]

We used the third option from the Tab. 5.1 for the majority of work and experiments on our Tomi2 platform. That means that we received 2560x720 image at 30 frames per second. This image had to be then sliced in half to receive typical resolution of 1280x720 for both left and right lens.

Part III

Implementation and experiments

Chapter 6

Implementation starting point

This thesis continues to work on research started by the Tomi2 project resulting in papers [1], and [2]. In these papers were implemented methods of predicting surface roughness and friction using self-supervised learning. We harvested the dataset using our Tomi2 platform, which recorded images and all the measurements from the sensors. The properties of the surface served as labels for the training of convolutional neural network and the images were annotated automatically.

Two ResNet50 [11] architecture models were trained on a dataset labeled automatically by cross-modal supervision. The experiments showed that both models are accurate visual predictors. The correlation coefficient between the visually predicted results and the true labels was 0.9 on the independent validation dataset for roughness and 0.98 for friction respectively. These methods provided automatic and objective road conditions assessment using a cheap and reliable alternative to manual data labeling, which enables to use this approach on large datasets [1].

This method provides single surface roughness/friction prediction for an entire image (1 meter in front of a vehicle) and lacks more detailed spatial resolution. This is particularly important for navigating on complex heterogeneous surfaces comprising various areas of different properties. Though the predictor outputs a single value, we wanted to see the generalization for the larger image area. A simple baseline to achieve that is by moving a sliding window over the original image and executing the CNN prediction for about 600 different areas. Those outputs are visualized in Fig. 6.1 and Fig. 6.2.

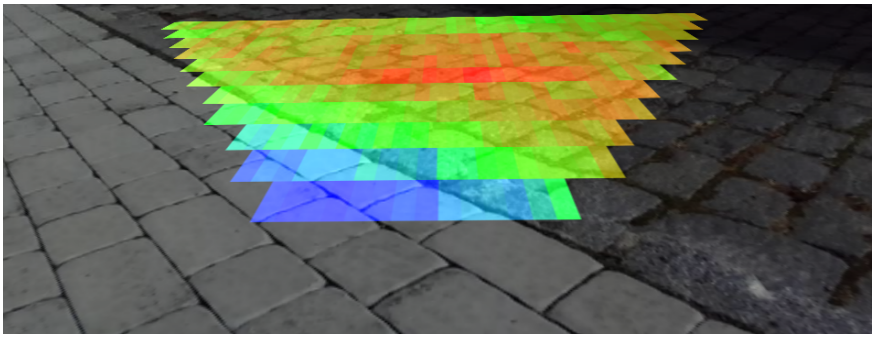


Figure 6.1: Color-coded surface roughness calculated by the trained CNN executed in a scanning window over the input image rectified to bird's-eye view, colorized by the estimated roughness, and finally back-projected to the raw camera image. [1]

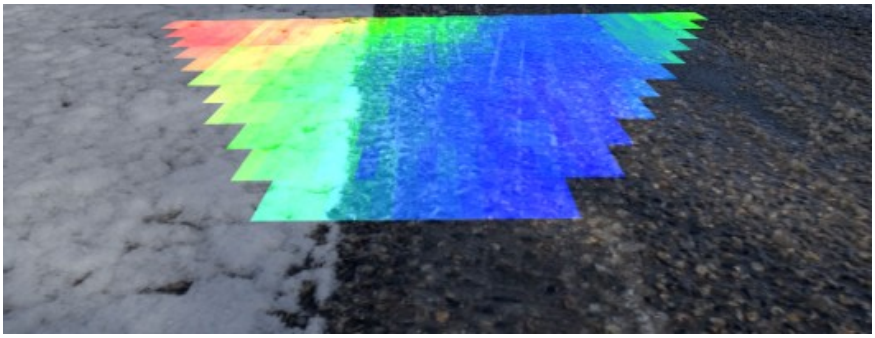


Figure 6.2: Color-coded friction of the surface found by a visual predictor that was trained only by self-supervision from vehicle response data, without any manual annotation. Colder colors encodes higher friction (wet tarmac), while warmer colors lower friction (snow). [2]

While this method of obtaining a coarse map of surface properties gives good results, it cannot be used in a system running in real time. It requires many hundreds of model executions, using an extended amount of processing time. Single prediction takes about fourty milisecond, several hundreds are therefore unacceptable for real-world world use. This thesis aims to increase the spatial resolution of the predictor. The output will be a map of local surface properties (both roughness and friction). The resulting model should provide the map using a single forward pass through the model, significantly reducing the time and resources needed for a prediction.

Chapter 7

Birds-eye transformation

In order to be able to provide a surface map in front of the vehicle , we need to somehow transform the original image to be rectangular and of given dimensions.

The process of orthorectification begins with capturing a raw ZED2 camera image. It is then rectified to a virtual bird's-eye view. The homography mapping is used to warp the raw image to a scene of the desired size. In the Sec. 7.1 and Sec.7.2 are described two main approaches to this problem.

7.1 Static homography

The static transformation orthographically rectifies the raw image to the bird's-eye view. The homography mapping is used to warp the image and receive a scene of a given size. The homography is defined by the corners of the scene rectangle and the corners of the rectified image. You can see the result of the static transformation in Fig. 7.1. The scene from the raw image on the left is transformed into the warped image representing 1.5 by 1.5 meters on the right.

(a) : Original image

(b) : Transformed image 1.5x1.5 metres in front of a vehicle

Figure 7.1: Static birds-eye transformation

The transformation was found by placing a planar rectangular object of known dimensions in front of a vehicle. The homography estimation was found using known corner points of the object source points and object points in orthorectified image as targets using getPerspectiveTransform() function from OpenCV library [41].

7.2 Dynamic birds-eye view

The main disadvantage of the static homography approach is the camera tilt due to instability of the vehicle body. The springs on the wheel base are not rigid and that can cause the mount of the ZED2 camera on top of the vehicle to move around in consequence of the acceleration of the car.

The idea of the dynamic birds-eye view transformation is to take into account the position of ZED2 camera. If we were able to estimate the immediate angle of the camera we could then adjust the homography to compensate camera tilting due to vehicle driving.

The first idea was to use an inertial measurement unit that is inside the ZED2 camera itself or some additional sensor. We would read its data in real time and use that information to create our transformation matrix. However, this approach has one limitation: we would not be able to differentiate from the pitch angle obtained whether the position of the camera has changed with respect to the vehicle body or whether the car platform is simply driving up or down the hill.

To solve the hill detection problem, we took advantage of the ZED2 camera to compute 3D pointcloud of the space from the stereo lenses. In the following sections will be described the computation of relative pitch angle of the camera with respect to ground plane detected in the pointcloud.

7.2.1 3D pointcloud

As described in Sec. 5.3.1 the ZED2 camera uses the camera stereo pair to compute a sparse 3D pointcloud of its surroundings. Figure 7.2 shows the raw stereo images on the top and on the bottom are computed respective pointclouds. Each point has three positional coordinates related to the left camera lens. In those pictures are depicted pointclouds with only one-tenth of maximal pointcloud density. This density allows us to perform robust ground plane detection and its normal vector computation without demanding much of limited memory and processor time.

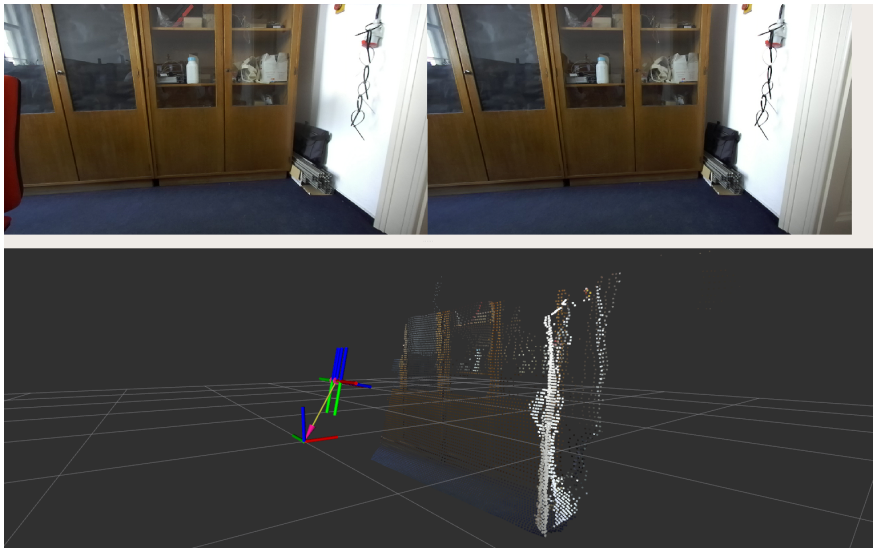


Figure 7.2: ZED2 3D pointcloud

7.2.2 Pointcloud filtering

When driving on an uneven surface, the vibrations can cause the final image to be a bit blurred. Those distortions (loss of high-frequency texture) in a stereo recording can cause some miscalculations when creating 3D pointcloud. You can see in Fig. 7.3 two groups of false points appearing above and below the actual scene.

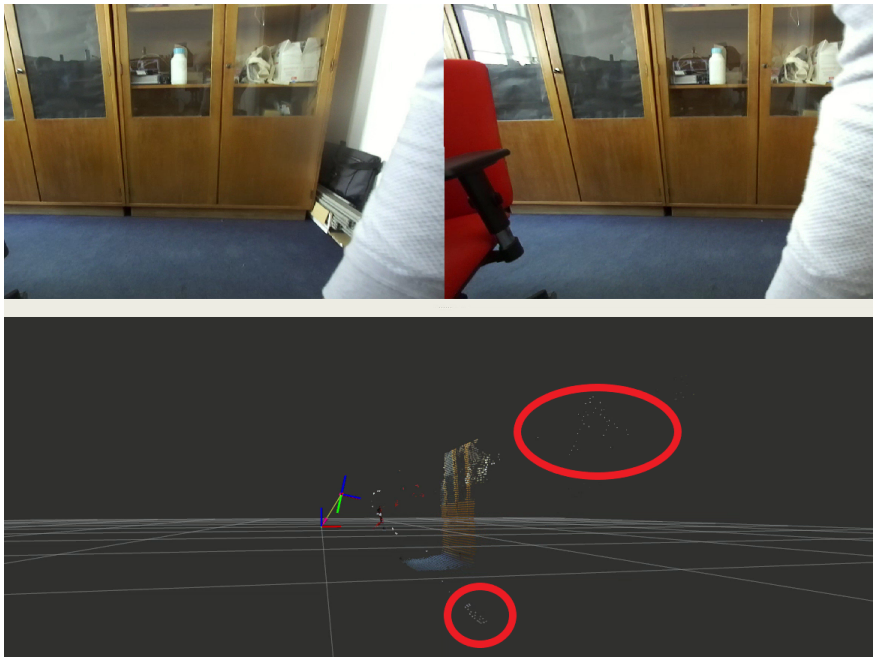


Figure 7.3: ZED2 3D pointcloud - miscalculations

We plan to estimate the normal vector of the ground plane using RANSAC (Sec.1) algorithm. RANSAC itself is a robust algorithm, but we can achieve faster and more reliable result by doing some preprocessing on the given pointcloud. The goal is to filter out obvious outliers from the scene and provide RANSAC with a small neighbourhood above the point with the minimal height in the scene. That leaves us mainly with the ground points and a much smaller number of points with different height than simply feeding RANSAC with the whole pointcloud scene.

The filtering itself is performed in the following way: first, we get rid of the points that lie below the height level of percentile of 2. That has been empirically found to be the best compromise between filtering out the outliers and at the same time keeping the maximum number of points that are present in the ground section of a pointcloud. After this operation, the points that have height difference bigger than ten centimeters from the lowest points of the scene are omitted. We have thus successfully obtained an area of pointcloud with the majority of ground points.

7.2.3 RANSAC ground plane detection

This section describes the process of computation of the normal vector of the ground plane detected in the filtered pointcloud from Sec.7.2.2. This data is fed into RANSAC algorithm implemented in the sklearn library [42]. A minimum number of samples chosen randomly from original data is set to fourty. A maximum number of iterations for random sample selection is set to two hundred. Squared loss is chosen as an error function. This algorithm returns the parameters of the plane.

Parameters of the plane are transformed into coordinates of the normal vector based in the vehicle platform base. Obtained coordinates are used to compute pitch angle of the camera with respect to the ground.

$$\theta = \operatorname{atan2}(z, x) \tag{7.1}$$

$$\operatorname{atan2}(y, x) = \begin{cases} \arctan(\frac{y}{x}) & \text{if } x > 0 \\ \arctan(\frac{y}{x}) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan(\frac{y}{x}) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ +\frac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0, \\ -\frac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{7.2}$$

In the Fig. 7.4 is visible comparison of the pitch angle obtained from the pointcloud (blue line) and the ground truth recorded by the ZED2 inertial measurement unit. We tilted the car from front to back changing the pitch angle. This plot shows the fact that we are able to use the angle computed using only the pointcloud from the camera and dynamically change the

birds-eye view transformation, which is described in the following chapter 7.2.4.

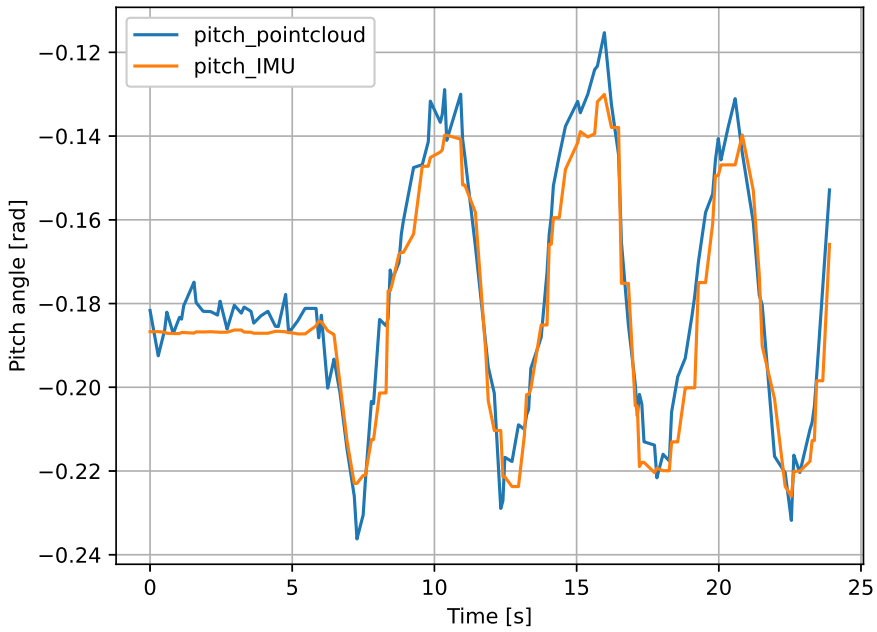


Figure 7.4: Comparison of pitch angle obtained from the ZED2 pointcloud and raw data from ZED2 IMU

7.2.4 Image transformation using obtained ground normal vector and camera intrinsic matrix

We first need to define the area in front of the vehicle we want to contain in the transformed image. Let's say that we want to send to map a segmented area of 2 by 2 meters. Then the scene points are defined as follows:

$$\mathbf{X} = \begin{bmatrix} 0.4 & 0.4 & 2.4 & 2.4 \\ 1 & -1 & 1 & -1 \\ -0.14 & -0.14 & -0.14 & -0.14 \\ 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{7.3}$$

The first row of the X matrix defines the distances in the x axis, which is defined in the direction of movement of the car. The reason for adding fourty centimeters to each point is that the main coordinate system is defined in the base of the vehicle, while the front wheels are fourty centimeters ahead. The second row defines two meters in the y axis. The third row shows that the base of the car is 14 cm above the ground. To change the bases of different coordinate systems, the points are defined in homogeneous coordinates.

The $\mathbf{T_{pitch}}$ matrix transforms the points between map and vehicle base coordinates using the computed pitch angle. Other euler angles are neglected, we are mostly affected by the forward and backward tilt of the vehicle body.

$$\mathbf{T_{pitch}}(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.4}$$

The following transformation matrix is $\mathbf{T}^{\mathbf{zed}}_{\mathbf{base}}$ which transforms points from vehicle base coordinate frame to the ZED2 left camera optical frame. This transformation was measured by hand and is stored in ROS2 transformation buffer.

$$\mathbf{X_{camera}} = \mathbf{T}^{\mathbf{zed}}_{\mathbf{base}}\mathbf{T_{pitch}}\mathbf{X} \tag{7.5}$$

The final step is to take transformed points and project them into camera image plane. That is done using camera intrinsic matrix:

$$\mathbf{K} = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.6}$$

, where f_x and f_y are focal lenghts, x_0, y_0 are principal point offsets and s is axis skew. To projects points obtained by the transformation to the image plane by the matrix K we first need to handle matrix computed in 7.5 (remove last row). Matrix K is saved by the manufacturer in ZED2 camera.

$$\mathbf{X_{pixels}} = \mathbf{K}\mathbf{X_{camera}} \tag{7.7}$$

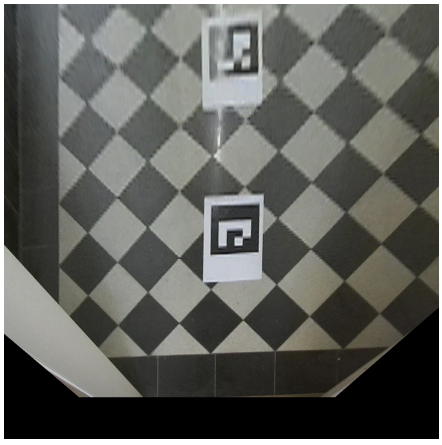
The matrix $\mathbf{X_{pixels}}$ computed in 7.7 contains pixel coordinates of the four points defining area in front of the vehicle. These pixel points are used to compute perspective transformation, which closes the whole dynamically computed birds-eye transformation.

7.3 Static and dynamic birdseye view comparison

We have performed an experiment for the purpose of comparing the static and dynamic transformations. The car body has been tilted back and forth around the y axis. The angles are seen in the Fig. 7.4. You can see steady states of the transformations in the Fig. 7.5. In the image are visible two ARUCO markers, the one in the bottom marks one-meter in front of car front axle and the top of the second one marks the two meter line.

Figure 7.5: 2x2 metres transformation

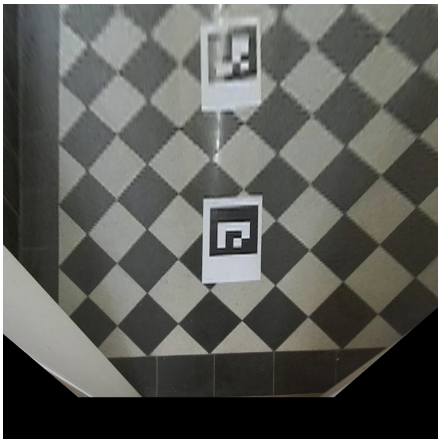
Visualization below shows a comparison of the two approaches to the image transformation. The Fig. 7.6, Fig. 7.7, Fig. 7.8 and 7.9 present two images taken at the exact same time each using different approach. It is clearly visible that dynamic transformation can react to the change of a vehicle body tilt, while static transformation loses track of the specified area and shows the surface way behind the two-meter mark.



(a) : Transformed image using dynamic transformation

(b) : Transformed image using static transformation

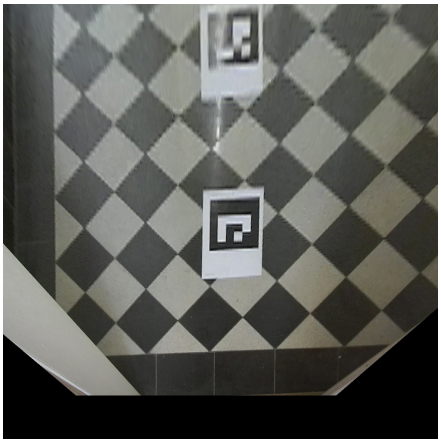
Figure 7.6: Comparison of static and dynamic transformation at time 8.0s



(a) : Transformed image using dynamic transformation

(b) : Transformed image using static transformation

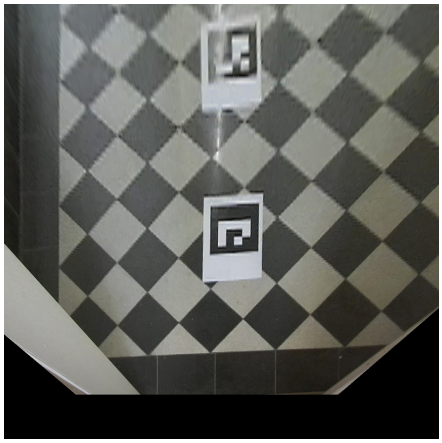
Figure 7.7: Comparison of static and dynamic transformation at time 8.7s



(a) : Transformed image using dynamic transformation

(b) : Transformed image using static transformation

Figure 7.8: Comparison of static and dynamic transformation at time 9.4s

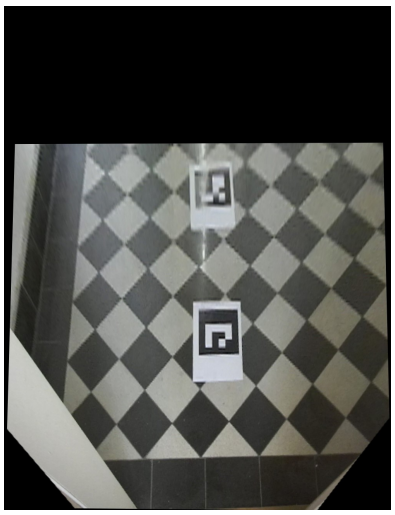


(a) : Transformed image using dynamic transformation

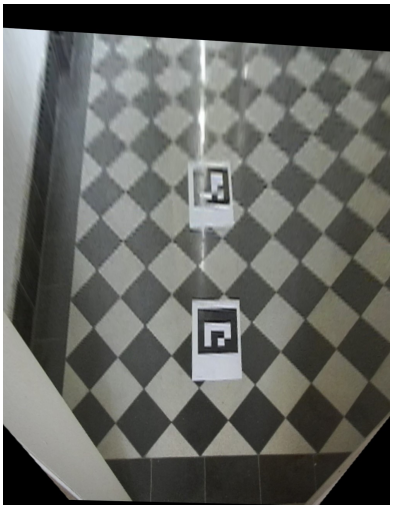
(b) : Transformed image using static transformation

Figure 7.9: Comparison of static and dynamic transformation at time 10.0s

Each rectified image is projected to the steady-state image of 2 meters by calculating the optical flow of good features [43] and resulting homography (using OpenCV [41]). The main reference points are in the same position of the image compared to the steady state. Fig. 7.10 when the rectified image shows a bigger area than desired. We can compute the area shown and difference from the desired size using images in Fig. 7.10.



(a) : Steady-state image



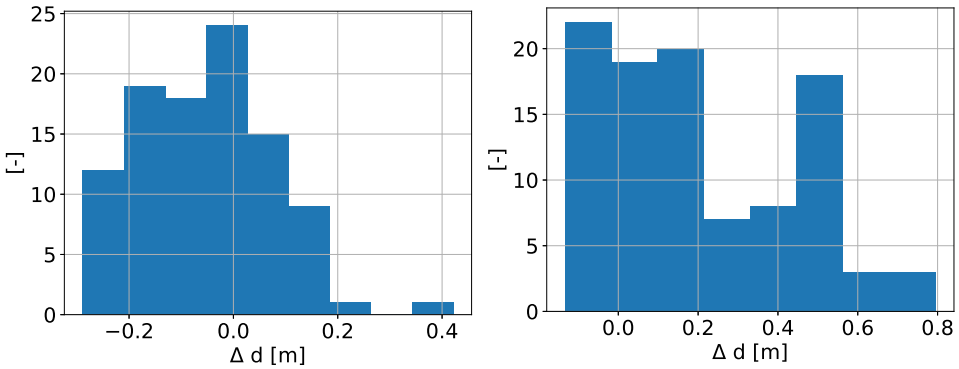
(b) : Projection of static transformation to the steady-state image

Figure 7.10: Comparison of steady-state area and area shown by rectification

Tab. 7.1 shows statistics from the experiment for both methods. It contains the mean value of distance shown in the rectified image and standard deviation from two-meter meter mark.

Transformation method	Mean [m]	Standard deviation [m]
Static	2.2	0.27
Dynamic	1.92	0.12

Table 7.1: Comparison of static and dynamic transformation



(a) : Histogram of differences from two-meter mark of dynamic correction

(b) : Histogram of differences from two-meter mark - static correction

Figure 7.11: Histograms comparing differences from two-meter mark of dynamic and static homography

The Fig. 7.11 shows the comparison of histograms of differences from the desired two-meter mark recorded during our experiment. From the histograms 7.11 and the Tab. 7.1 is visible that dynamic transformation is much more stable, while the static transformation has a higher deviation from the desired two meters.

Chapter 8

Dataset collection

8.1 Synthetic images

This section describes the collection of the semi-synthetic dataset that contains label for each pixel of a given rectified image. The camera recordings and sensor measurements were collected during rides (about twelve hours) on several different surfaces. The dataset contains information about asphalt, gravel, grass, snow, cobblestones, and tiled pavement.

8.1.1 Sparse spatial information problem

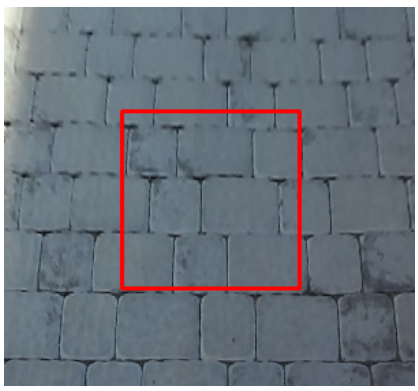
The goal of this model is to obtain a prediction map of local surgace properties for the whole surface area in front of a vehicle. The problem of the original method described in Sec. 6 is the sparsity of the surface property information in the image. Measurements on the Tomi2 platform are recorded at 100 Hz. If we consider the two by two meters rectified area and 10 $\frac{m}{s}$ vehicle speed, there are only 20 sensoric measurements in that window. The new method aims to provide information about the surface for every pixel of the image.

We created a synthetic dataset of images resembling natural images that would possess label for each pixel. We can rely on the assumption that each drive was performed on a single surface. That excludes the possibility of having two different surfaces in rectified images.

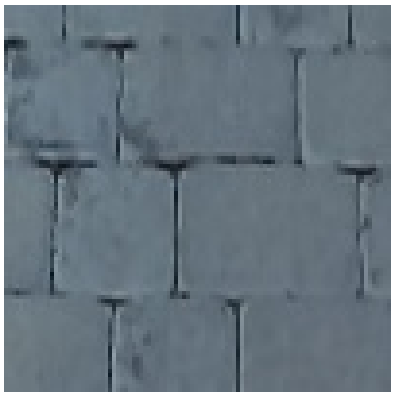
The synthetic image creation starts by cutting a small patch from each rectified image. We choose the area around the one-meter mark in front of a vehicle. One of the roughness or friction measurements that lies closest to this measurement region is taken as a label for this patch. We then cut the area of 40 by 40 centimeters around this pixel using the same label for the whole patch. Fig. 8.1 shows the original image for the patch cutting.

Figure 8.1: Original camera imaged to be patched

Fig. 8.2 shows the process of the patch cutting. Original camera image 8.1 is transformed as shown in (a). Then we cut the measurement region resulting in a final patch (b).



(a) : Image rectification



(b) : Final patch

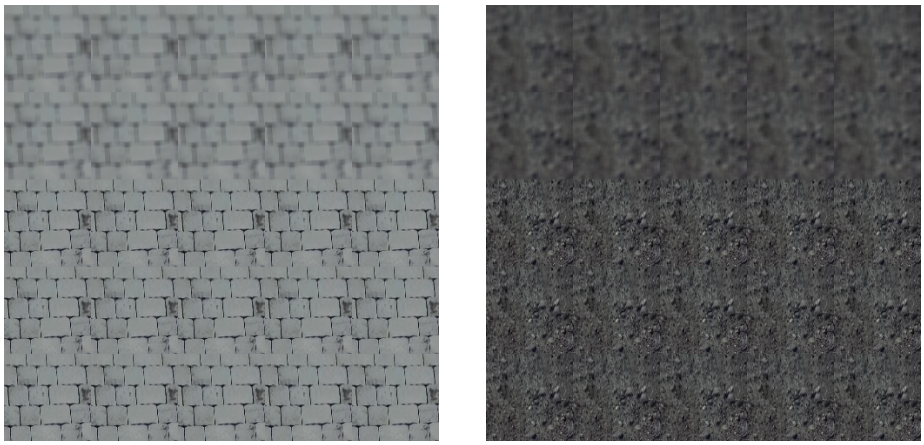
Figure 8.2: Patch creation

8.1.2 Patch concatenation and blurring

Once we have the patches for all the images, we can create the synthetic images. We imitate the natural images as much as possible to achieve the best accuracy on the non-synthetic validation dataset. We decided to create each one of the synthetic images using two random patches. Each of those patches is from a different ride. The synthetic image has square dimensions of 224 by 224 pixels, which are the dimensions our CNN models will expect as inputs.

Once we picked the two patches, we create two different images, that will

be later on fused together, using only those patches.



(a) : Image created using the first patch **(b) :** Image created using the second patch

Figure 8.3: Patch images for fusion

For the fusion of the images is needed a binary mask. We do not want to feed the network with the repetitive patterns, so the mask will be unique for each synthetic image. We start with an initial seed randomly located in the binary mask. Starting from this point we generate 300 other points, each randomly in the neighbourhood of 20 pixels of the previous. This cloud of points is then dilated using the morphological operation and ellipse kernel to obtain the final mask. The process is shown in Fig. 8.4.

(a) : Cloud of random points **(b) :** Final mask

Figure 8.4: Random binary mask creation

The two images are finally fused together using the obtained binary mask in Fig. 8.5. There are added black corners to the final image. This element is present in transformed images. The black corners represent the invisible part of the area in front of the vehicle for the camera. The top of the image

is blurred by the averaging (using cv2.blur()). It emulates the degradation of the transformation in the distance due to camera resolution.

Figure 8.5: A sample of a synthetic image from the dataset

8.1.3 Dataset distributions

Two datasets of 10 000 synthetic images with pixel-wise labels were created, both for surface roughness and surface friction. The synthetic dataset consists of training part (80%) and validation part (20%). Test data are described in Sec. 8.2. In the Fig.8.6 and Fig.8.7 are the distributions of datasets for the friction and for the roughness respectively.

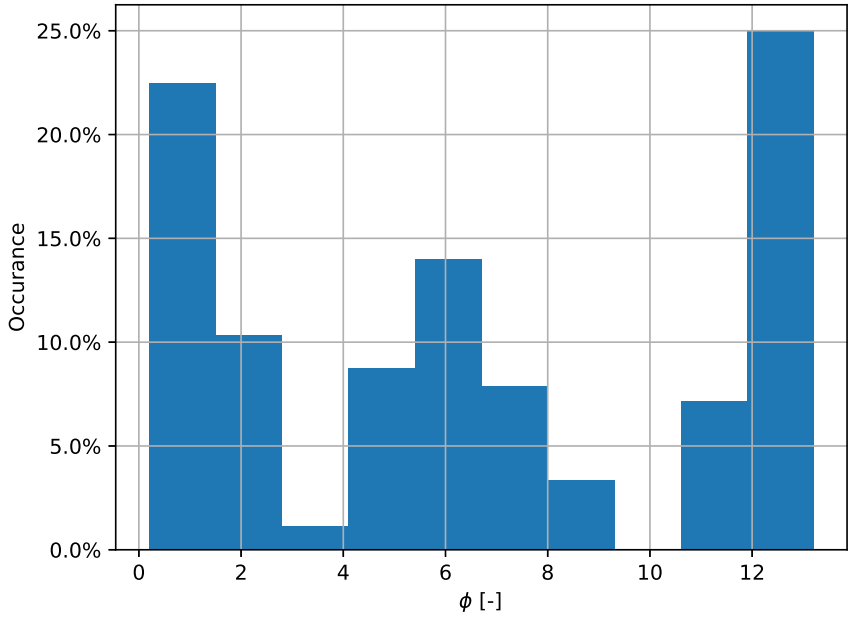


Figure 8.6: Histogram of surface friction synthetic dataset

The Tab. 8.1 shows the typical value of fricion for different surfaces. It is necessary to point out that those are values measured on a specific conditions. These values can change depending on the weather (mostly rain). We would ideally need a datasets containing surfaces driven in different weather conditions and mainly different lighting conditions. The differences in the lighting (direct sun, shadows) make the prediction problem especially challenging.

Surface	**Typical value of friction ϕ**
Grass	2.5
Ice/snow	1.4
Asphalt	12.1
Interlocking pavement	7.2
Cobble pavement	7.9
Hard gravel	10.9
Wet cobblestones	3.0

Table 8.1: Typical values of surface friction for different surfaces

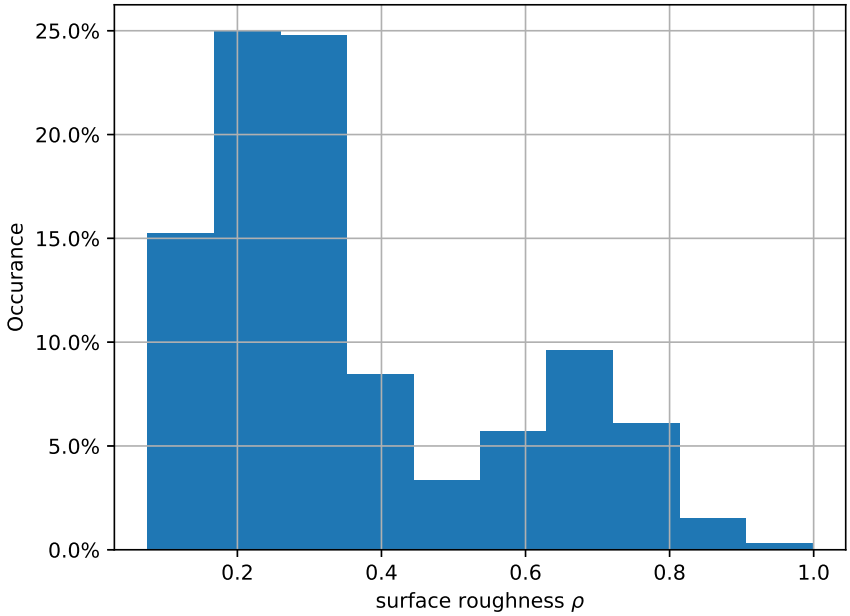


Figure 8.7: Histogram of surface roughness synthetic dataset

Tab. 8.2 shows typical values of roughness for different surfaces. The value of surface roughness does not depent much on the weather conditions. The lighting conditions still play a big role in the training proccess.

Surface	Typical value of roughness ρ
Grass	0.3
Asphalt	0.15
Tiled pavement	0.2
Cobble pavement	0.66
Gravel	0.25
Cobblestones	0.75
Ice/snow	0.3

Table 8.2: Typical values of surface roughness for different surfaces

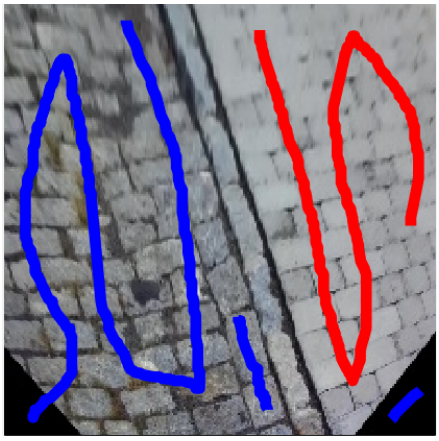
8.2 Test dataset

It is necessary to do the final testing of the model on non-synthetic images to measure the real world performance. This is done by manually annotating the surface properties. That was done using graph-cut segmentation and Matlab GUI provided in B4M33DZO course. The algorithms used by the GUI were programmed in seminars.

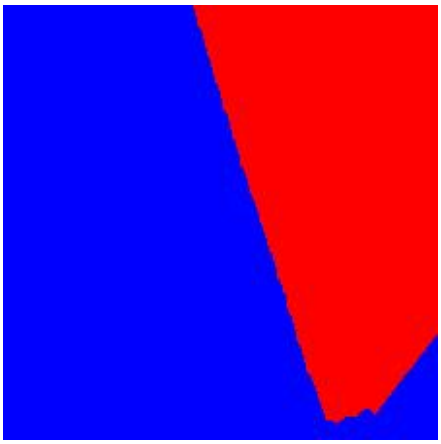
The algorithm uses Gaussian mixture model (GMM) based segmentation. The user input is used as information about the foreground and background.

It is visible in Fig. 8.9 (a). The red and blue pixels are used to estimate probability distribution functions of foreground and background pixels modeled by GMMs. Individual components are formed by multivariate normal distributions. The pixels are then coloured respectively by the distributions. The smoothness is ensured by finding minimum cut (maximum flow) in a graph constructed from the pixels [44].

Figure 8.8: Original image for validation partition of dataset



(a) : Birdseye transformation of 8.8 with manual input



(b) : Result of graphcut segmentation

Figure 8.9: Image segmentation for validation dataset

The red and blue colors are assigned the values of the surface properties. The test datasets of fifteen images were created both for surface roughness and friction.

Chapter 9

Pixel-wise regression

9.1 Model training

We trained two different models for surface roughness and surface fricion respectively. The U-net architecture [13] was chosen for both models. This architecture has been developed for biomedical image segmentation and is often used to perform semantic segmentation. The model accepts images of 224 by 224 pixels. The original architecture is described in Sec. 4.5.2.

The models were trained on a GPU server (Cantor server available for the students by the Department of Cybernetics at CTU FEE). This server has 16 cores/32 threads CPU, 256 GB of RAM, 500GB SSD internal storage, and eight Nvidia GTX 1080 Tis. These GPUs have 11 GB of memory and are crucial for the model training. The CUDA architecture (3584 cuda cores) is compatible with the PyTorch module which was used for the model implementation.

Mean square error loss function was used for surface roughness and mean absolute error for surface fricion. The loss is computed on the output segmentation map of the Unet (224 by 224 pixels) and the image label of same dimensions. Network weights were found by ADAM optimizer [45].

Models for both surface roughness and friction were trained on a synthetic datsets of 10000 samples. Evaluation of model on validation part of the dataset was performed after every training epoch. The model with the smallest error was saved for evaluation on the test dataset. Training and validation parts of the datasets consist of the synthetic images. The testing part is then performed on manually annotated non-synthetic images. Training settings are shown in Tab. 9.1.

Surface property	Surface roughness 3.1	Surface fricion
Learning rate	0.01	0.01
Image size	224 px by 224 px	224 px by 224 px
Batch size	64	64
Loss function	MSE 4.6	MAE 4.7
Number of epochs	50	60

Table 9.1: Training settings for models

9.2 Surface roughness evaluation

A model trained on the synthetic data was used to predict segmentation maps for the images from the test dataset of real images. The model achieved mean square error of 0.016, that means mean absolute error of 0.126 for every pixel. That is accuracy that allows the model to distinguish between the surfaces. Folowing images(Fig.9.1 and Fig. 9.2) show the predictions of the surface roughness map for the manually annotated data.

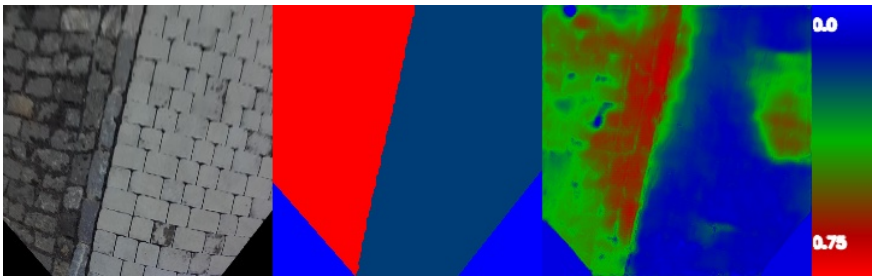


Figure 9.1: Example of roughness map on cobbles and interlocking pavement (Original image -> Manual annotation -> Model prediction -> Value bar)

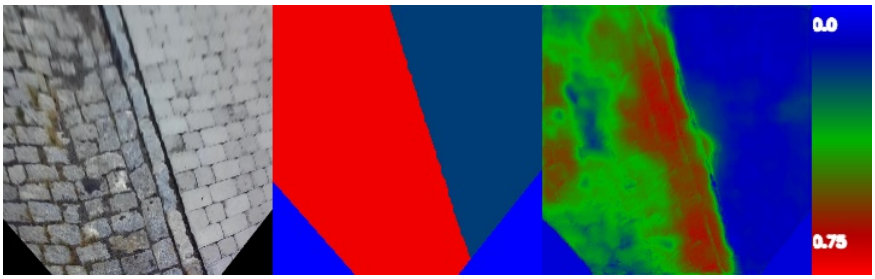


Figure 9.2: Example of roughness map on cobbles and interlocking pavement (Original image -> Manual annotation -> Model prediction -> Value bar)

Fig. 9.3 contains an original image with the grass covered partialy with snow. That causes a little confusion for the predictor, because such a situation is not present in a training dataset. Training data contain images of grass

recorded in summer. The model still manages to predict reasonable values for this surface.

Figure 9.3: Example of roughness segmentation map on asphalt, and grass with snow (Original image -> Manual annotation -> Model prediction -> Value bar)

9.3 Surface friction evaluation

The predictor trained on data with surface friction as a label manages to achieve a mean absolute error of 1.49. The figures below (Fig.9.4, Fig.9.5, Fig. 9.6 and Fig.9.7) show the outputs of the U-Net model. Better performance of the predictor could probably be achieved by enlarging our dataset by images from different lighting conditions. Artifacts as shadows can cause inaccuracies when unseen in training.

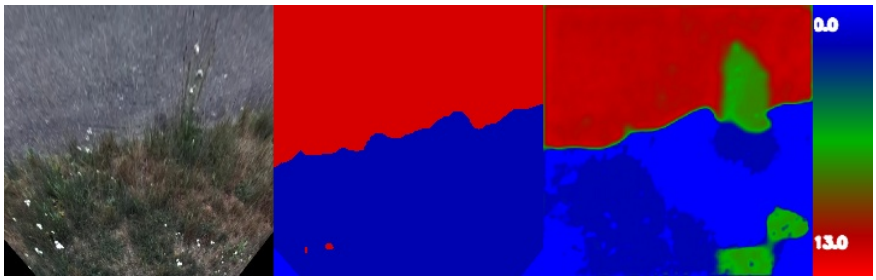


Figure 9.4: Example of friction map on asphalt and grass (Original image -> Manual annotation -> Model prediction -> Value bar)

Figure 9.5: Example of friction map on cobbles tiled pavement (Original image -> Manual annotation -> Model prediction -> Value bar)

The orange line in the tiled pavement in Fig. 9.6 was not seen in a training dataset and the predictor misinterprets it as a different surface.

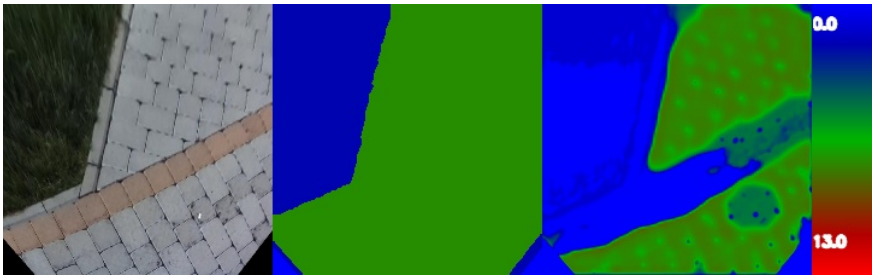


Figure 9.6: Example of friction map on cobbles and tiled pavement with unknown artifact (Original image -> Manual annotation -> Model prediction -> Value bar)

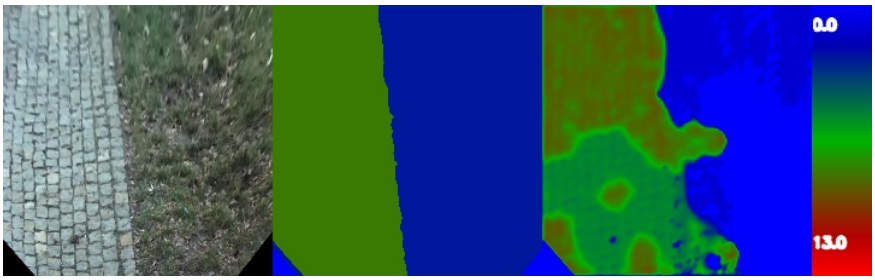


Figure 9.7: Example of friction map on cobble pavement and grass (Original image -> Manual annotation -> Model prediction -> Value bar)

Fig. 9.8 shows the effect of imperfections in the original image. There is a dirt or drop of water on the lens and the model predicts very different values for this area.

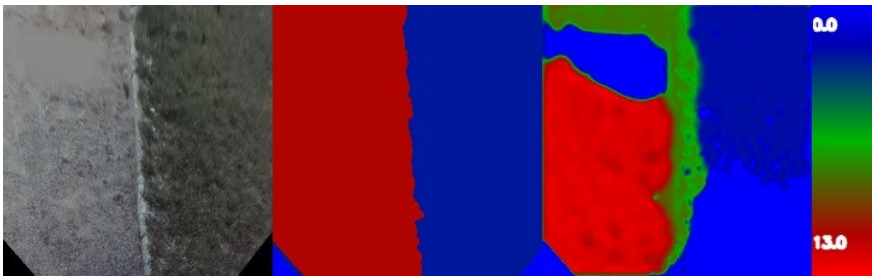


Figure 9.8: Example of friction segmentation map on gravel and grass (Original image -> Manual annotation -> Model prediction -> Value bar)

Chapter 10

Grad-CAM segmentation

The second method of obtaining the prediction map of local surface properties will be based on the visual explanations from deep networks via gradient-based localization Grad-CAM [37]. Grad-CAM is used for better visualization of image patterns of classes that network is sensitive to as described in Sec. 4.6. The following section describes the modification of this weakly supervised method to the purpose of prediction of local surface maps.

10.1 Grad-CAM pipeline

To make Grad-CAM work, we have to make adjustments to the models described in Sec. 6. We change the final layers of Resnet50 models trained to predict the single label of surface property. The final fully connected layer is retrained to predict the probability of artificial classes. Original continuous labels will be assigned to these classes based on a lookup table.

Artificial classes	Surface roughness	Surface fricion
Class 1	$\rho < 0.25$	$\phi < 3$
Class 2	$0.25 \geq \rho < 0.5$	$3 \geq \phi < 8$
Class 3	$\rho \geq 0.5$	$\phi \geq 8$

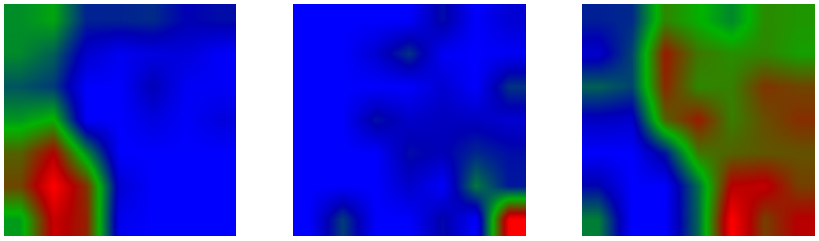
Table 10.1: GradCAM classes lookup table

The final fully connected layer is changed to softmax (three outputs), predicting a probability of the input belonging to the classes defined in Tab. 10.1. Both models are retrained for ten epochs using the same settings described in papers [1], and [2]. Only the final layer of the models is modified, the convolutional layers stay unchanged.

Our algorithm is demonstrated in Fig. 10.1. The image with tiled pavement and cobblestones serves as an input for our modified model predicting surface roughness. The output of the model (the probabilities of belonging to classes)

are used to find the gradients of each class with respect to the latest activation map in the model using Eq. (4.10). The result is used in Eq. (4.11) which outputs the heatmaps for each class as seen in Fig. 10.2.

Figure 10.1: Input image for GradCAM



(a) : GradCAM output class 1

(b) : GradCAM output class 2

(c) : GradCAM output class 3

Figure 10.2: GradCAM outputs for three classes

Heatmaps from Fig. 10.2 are two dimensional matrices and are caled $\mathbf{g_1}, \mathbf{g_2}$ and $\mathbf{g_3}$. We will perform a pixel-wise normalization. That means that the sumation of values at given matrix position from all class probability maps is equal to one.

$$\alpha(x, y) = \sum_{c=1}^{3} \mathbf{g_c}(x, y) \tag{10.1}$$

$$\tilde{\mathbf{g}} = \frac{\mathbf{g}(x, y)}{\alpha(x, y)} \tag{10.2}$$

The final output is computed as:

$$\hat{\mathbf{y}}(x, y) = \sum_{c=1}^{3} \tilde{\mathbf{g}}_\mathbf{c}(x, y)\overline{y_c}, \tag{10.3}$$

where the $\overline{y_c}$ is expected value of the property for the classes defined in Tab. 10.1. The output is shown in Fig. 10.3 alongside the original image and manual annotation.

10.2 Surface roughness predictions

The outputs of the model predicting surface roughness are visible in Fig. 10.3 and Fig. 10.4. This model achieves a mean absolute error of 0.207 on the test dataset. The disadvantage is the low resolution of the original output $\hat{\mathbf{y}}(x, y)$. The size of the activation map of the last convolutional layer determines the size of the model output, which is seven by seven two dimensional matrix. This output is then resized to the dimensions of the input image, but the original information is still only 49 values. This limitation can cause imperfections in following the transitions between different surfaces and overall error with respect to the manual annotation.

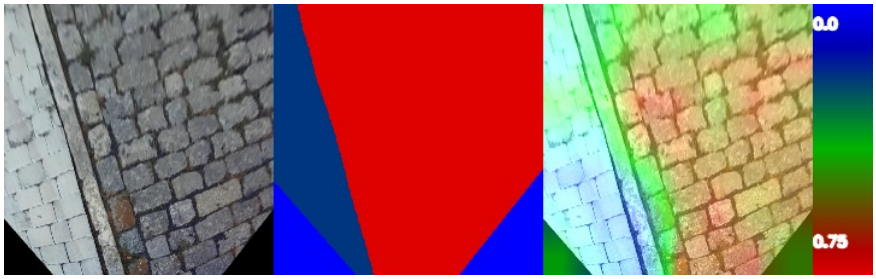


Figure 10.3: Segmentation map created by GradCAM (Original image -> Manual annotation -> Model prediction -> Value bar)

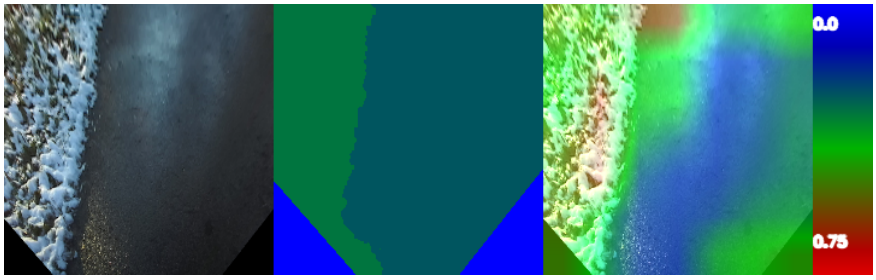


Figure 10.4: GradCAM map for asphalt and grass with snow (Original image -> Manual annotation -> Model prediction -> Value bar)

10.3 Surface friction predictions

Outputs for surface friction are in Fig. 10.5, Fig. 10.6 and Fig. 10.7. The friction model achieves a mean absolute error of 4.3 on test data. The same limitations apply to both the friction and roughness. The improvement of

this method could be increasing the number of artificial classes. Possible future improvements of the methods will be discussed in future work.

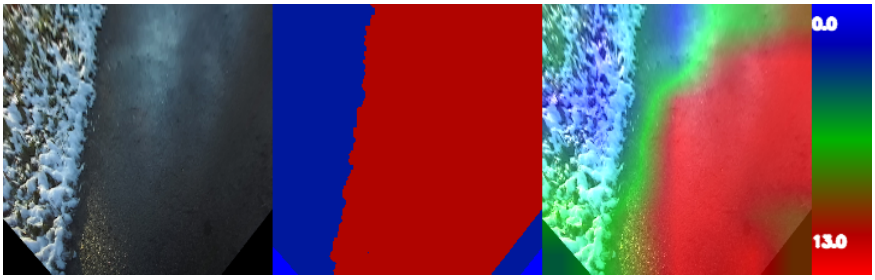


Figure 10.5: GradCAM map for asphalt friction (Original image -> Manual annotation -> Model prediction -> Value bar)

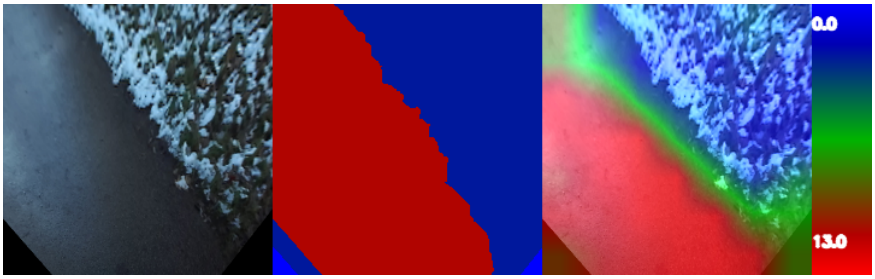


Figure 10.6: GradCAM map for asphalt and grass friction (Original image -> Manual annotation -> Model prediction -> Value bar)

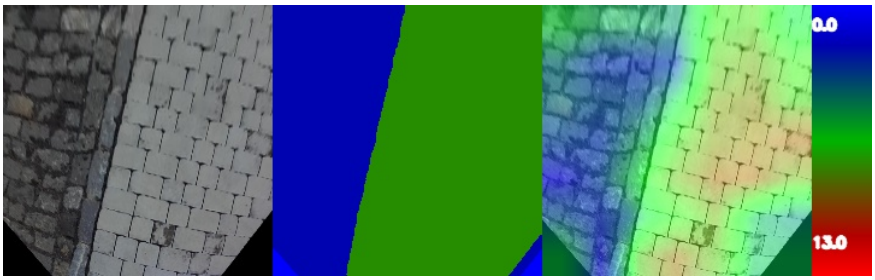


Figure 10.7: GradCAM map for cobblestones and pavement friction (Original image -> Manual annotation -> Model prediction -> Value bar)

Chapter 11

Results

In the Tab. 11.1 are shown results of four models predicting surface properties. Pixel-wise regression model using UNet [13] architecture predicting surface roughness achieves mean absolute error of 0.126. The same model predicting surface friction achieves a mean absolute error of 1.49. The significant difference in the values is not caused by the major differences in model qualities. The main cause is the different ranges of those two surface properties. While the surface roughness maximum values are about 0.8, the values of friction go up to 13. The error with respect to the values range is similar for both properties. The error on the semi-synthetic validation dataset is about seventy-five percent of the error on the natural images. That is probably due to the imperfections in emulating real-world images.

The approach using visual explanations from deep networks via gradient-based localization [37] records about a double of errors in comparison with dedicated segmentation architecture. The error of this method is partly influenced by the resolution of the output segmentation map, that has to be upsampled to match the original image. A number of chosen bins (artificial classes) is another factor for the performance. Using more than three classes could probably lead to a better predictions. The original idea was developed for making human-readable explanations of the model behaviour, so proving that it can be used for pixel-wise regression is a good result.

Model	Mean square error	Mean absolute error
Roughness UNet	0.0016	0.126
Roughness Grad-CAM	0.043	0.207
Friction UNet	-	1.49
Friction Grad-CAM	-	4.3

Table 11.1: Models results

Chapter 12

Conclusions

The first part of this thesis contains a survey of related methods dealing with surface properties description and self-supervised learning. The first part also contains a definition of two properties used to define surfaces: surface roughness and surface friction. It is followed by a theoretical background, that is composed of machine learning methods used in this thesis. More in-depth descriptions of convolutional neural networks and visual explanations from deep networks via gradient-based localization are provided.

The implementation part begins with the description of previous work of predicting a single label of the surface property for an input image, that is used as a starting point for the implementation of this thesis. Then we describe the creation of the synthetic dataset, that contains a full pixel-wise label for each image, which is needed for the first method of predicting full surface property maps in front of the vehicle.

Then is presented the rectification of the image taken by the camera to obtain a birds-eye view of a rectangular region of given dimensions in front of a vehicle. The first approach uses static homography to transform the original image into a birdseye view. The second approach uses the three-dimensional pointcloud provided by the ZED2 camera mounted on our platform to estimate the pitch angle between camera lenses and the ground plane detected in pointcloud. This information allows the algorithm to adjust the birds-eye transformation depending on the vehicle's current state. The second approach compensating the pitch angle dynamically, outperforms the static rectification in a real-world experiment. This provides a robust method of obtaining the area of given dimensions in front of a vehicle, even though the vehicle body is not rigid and the tilt of the camera changes.

The first method, that provides the pixel-wise surface property map for both roughness and friction, is explained. It describes the architecture and training settings. The models are trained on the previously created semi-synthetic datasets. Output segmentation maps for non-synthetic real-world images are shown in this chapter and are compared to manual annotations.

The second method providing a surface segmentation map is based on a visual explanation of deep networks via gradient-based localization (Grad-CAM) introduced in [37]. This approach is used for classification networks, so the necessary adjustments to our model are described. Model outputs probabilities of a surface belonging to one of three classes instead of outputing single regression values. These probabilities are used as weights and fused together to a final property map. Despite this method's limitations, it provides promising results for a model that is trained on automatically labeled data with just a single information for a whole image.

12.1 Future work

The main future improvement would likely be a creation of a more complex and larger dataset. Convolutional networks are sensitive to lighting conditions. The ideal state would be to have images for each surface during different day times and weather conditions. The colour of some surfaces can change significantly depending whether it rained or not. The automated system for image recording was developed, so it would not be necessary to rely on humans to do the annotation manually.

The first method could be improved by more advanced data augmentation techniques in the creation of the synthetic dataset. The second method based on Grad-CAM could be improved by extending the number of artificial classes. The current state provides only a coarse discretization for surface properties, so more classes could improve model accuracy. The modifiability of original Grad-CAM method to a problem of surface pixel-wise regression was proven and it could be further improved and researched.

A method of training convolutional networks on non-synthetic data, while using only parts of the image, where the sensor measurements are available, could be worth a try. This approach would have to reconstruct vehicle trajectory in the rectified image, and that relies on precise odometry, but the process of emulating real-world images by synthetic ones could be omitted.

Appendices

Appendix A

Bibliography

[1] J. Cech, T. Hanis, A. Konopisky, T. Rurtle, J. Svancar, and T. Twardzik, "Self-supervised learning of camera-based drivable surface roughness," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1319–1325, 2021.

[2] D. Vosahlik, J. Cech, T. Hanis, A. Konopisky, T. Rurtle, J. Svancar, and T. Twardzik, "Self-supervised learning of camera-based drivable surface friction," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pp. 2773–2780, 2021.

[3] J.-M. Dai, T.-A. J. Liu, and H.-Y. Lin, "Road surface detection and recognition for route recommendation," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 121–126, 2017.

[4] M. Bahnik, D. Filyo, D. Pekarek, M. Vlasimsky, J. Cech, T. Hanis, and M. Hromcik, "Visually assisted anti-lock braking system," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1219–1225, 2020.

[5] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture." arXiv 1411.4734, 2014. https://arxiv.org/abs/1411.4734.

[6] Mercedes Benz Youtube channel, "Mercedes-Benz MAGIC BODY CONTROL | S-Class," 2015. `https://www.youtube.com/watch?v=ScpgI1w5F6A`.

[7] Audi technology portal, "Audi A8 – Predictive active suspension," 2021. `https://www.audi-technology-portal.de/en/chassis/suspension-control-systems/audi-s8-predictive-active-suspension`.

[8] "Auto.cz," 2021. https://www.auto.cz/jezdili-jsme-specialni-octavii-s-peti-koly-k-cemu-slouzi-kolecko-navic-139047.

[9] K. Zimmermann, "Vision for robotics lectures," 2021. Czech Technical University in Prague.

[10] F.-F. Li, "Convolutional neural networks for visual recognition," 2017. Computer Science Department at Stanford University.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[12] S. Wang, X. Xia, L. Ye, and B. Yang, "Automatic detection and classification of steel surface defect using deep convolutional neural networks," *Metals*, vol. 11, p. 388, 02 2021.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation." arXiv 1505.054597, 2015.

[14] "Stereolabs website." https://www.stereolabs.com/zed-2/.

[15] M. Cao, Q.-V. Tran, N.-M. Nguyen, and K.-T. Chang, "Survey on performance of deep learning models for detecting road damages using multiple dashcam image resources," *Adv. Eng. Informatics*, vol. 46, p. 101182, 2020.

[16] D. Lydon, S. Taylor, M. Lydon, and J. Early, "A review of vision based methods for pothole detection and road profile analysis," in *Civil Engineering Research Association of Ireland (CERI 2020)*, Aug. 2020. CERI 2020 : Civil Engineering Research Ireland ; Conference date: 27-08-2020 Through 28-08-2020.

[17] V. Slavkovikj, S. Verstockt, W. De Neve, S. Hoecke, and R. Van de Walle, "Image-based road type classification," pp. 2359–2364, 08 2014. International Conference on Pattern Recognition.

[18] A. Riid, D. L. Manna, and S. Astapov, "Image-based pavement type classification with convolutional neural networks," in *2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES)*, pp. 55–60, 2020.

[19] D. Lee, S. Kim, H. Lee, C. Chung, and W.-Y. Kim, *Paved and Unpaved Road Segmentation Using Deep Neural Network*, pp. 20–28. 03 2020.

[20] E. Šabanovič, V. Žuraulis, O. Prentkovskis, and V. Skrickij, "Identification of road-surface type using deep neural networks for friction coefficient estimation," *Sensors*, vol. 20, p. 612, 01 2020.

[21] P. G. Brindha, M. Kavinraj, P. Manivasakam, and P. Prasanth, "Brain tumor detection from MRI images using deep learning techniques," *IOP Conference Series: Materials Science and Engineering*, vol. 1055, p. 012115, feb 2021.

[22] C. Chun and S.-K. Ryu, "Road surface damage detection using fully convolutional neural networks and semi-supervised learning," *Sensors*, vol. 19, no. 24, 2019.

[23] T. Lee, C. Chun, and S.-K. Ryu, "Detection of road-surface anomalies using a smartphone camera and accelerometer," *Sensors*, vol. 21, p. 561, 01 2021.

[24] M. Shneier, T. Chang, T. Hong, W. Shackleford, R. Bostelman, and J. Albus, "Learning traversability models for autonomous mobile vehicles," *Auton. Robots*, vol. 24, pp. 69–86, 11 2008.

[25] A. Yunusov, S. Eshkabilov, D. Riskaliev, and N. Abdukarimov, "Estimation and evaluation of road roughness via different tools and methods," 07 2019.

[26] H. Bello Salau, A. Onumanyi, A. Aibinu, L. Onwuka, J. Dukiya, and H. Ohize, "A survey of accelerometer-based techniques for road anomalies detection and characterization," vol. 3, pp. 8 – 20, 03 2019.

[27] A. Gonzalez, E. OBrien, Y.-Y. Li, and K. Cashell, "The use of vehicle acceleration measurements to estimate road roughness," *Vehicle System Dynamics - VEH SYST DYN*, vol. 46, pp. 483–499, 06 2008.

[28] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge.," *J. Field Robotics*, vol. 23, pp. 661–692, 01 2006.

[29] W. Kleine-Beek, "Runway friction characteristics measurement and aircraft braking RuFAB, volume 3: Functional friction," tech. rep., European Union Aviation Safety Agency, 2010.

[30] H. B. Pacejka and I. J. M. Besselink, *Tire and vehicle dynamics*. Amsterdam: Elsevier, third edition ed., 2012.

[31] P. Pošík, "Artificial intelligence lectures," 2021. Czech Technical University in Prague, FEE.

[32] X. Jin and J. Han, *K-Means Clustering*, pp. 563–564. Boston, MA: Springer US, 2010.

[33] D. Dzyabura and H. Yoganarasimhan, "Machine learning and marketing," in *Handbook of Marketing Analytics*, Edward Elgar Publishing, 2018.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[35] Z. Qawaqneh, A. A. Mallouh, and B. D. Barkana, "Deep convolutional neural network for age estimation based on VGG-face model," 2017.

[36] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, pp. 611 – 629, 2018.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, pp. 336–359, oct 2019.

[38] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, p. 381–395, jun 1981.

[39] W. M. Y. Wan Bejuri, M. Mohamad, and R. Z. Raja Mohd Radzi, "Emergency rescue localization (erl) using gps, wireless lan and camera," *International Journal of Software Engineering and Its Applications*, vol. 9, pp. 217–232, 09 2015.

[40] S. Choi, T. Kim, and W. Yu, "Performance evaluation of ransac family," *British Machine Vision Conference*, vol. 24, pp. 1–12, 01 2009.

[41] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, June 1994.

[44] V. Hlavac, "B4M33DZO - Graph-based image segmentation," 2020. FEE CTU Prague.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization." arXiv.1412.6980, 2014.