Vision-based Terrain Segmentation and Roughness Estimation

Application on the CENTAURO Robot

Vivekanandan Suryamurthy

Master of Science Thesis

Vision-based Terrain Segmentation and Roughness Estimation

Application on the CENTAURO Robot

MASTER OF SCIENCE THESIS

For the degree of Master of Science in Mechanical Engineering at
Delft University of Technology

Vivekanandan Suryamurthy

August 29, 2018.

Acknowledgement

It has been an amazing and fruitful experience working at the Department of Advanced Robotics, Istituto Italiano Tecnologia (IIT). I am extremely grateful to Dr. Dimitrios Kanoulas for being the best supervisor one could hope for. He has been an immense support right from day one and instrumental in the success of the project which has been submitted to Humanoids 2018. I would like to thank Dr. Nikos Tsagarakis for giving me the opportunity to work on CENTAURO EU project. My deepest appreciation to Vignesh Sushrutha Raghavan for providing valuable guidance during the course of the thesis.

My sincerest appreciation to Prof. dr. ir. Martijn Wisse for providing valuable suggestions and devoting his time to evaluate my work. I would like to extend my thanks to Dr. ir. Julian Kooij and Thomas Hehn for being part of my thesis committee.

A huge thanks to my family members who have been my constant support system, encouraging and pushing me to achieve my dreams. Finally, a special thanks to my friends in Delft for cheering me on and always being there for me.

Contents

1

Paper

Vision-based Terrain Segmentation and Roughness Estimation, with Convolutional Neural Networks: Application on the CENTAURO Robot

Vivekanandan Suryamurthy[1], Dimitrios Kanoulas[2], Arturo Laurenzi[2], and Nikos G. Tsagarakis[2]

Abstract— **Intelligent terrain perception for search-and-rescue robotic applications, requires a high-level understanding of both the terrain type and its chief physical characteristics. Roughness is one such important terrain property, since it could play a key role in robot control/planning strategies, while navigating in an unknown environment. In this paper, we present a single deep neural network architecture that predicts the pixel-wise terrain labels (i.e., sand, stone, wood, metal, road/sidewalk, and grass) and regresses their roughness from an input RGB image. Our approach, inspired by human analogy, leverages the basic image feature space from a pre-trained network (SegNet) to estimate the roughness. We experimentally validate our approach in real-world images, using RGB cameras. Moreover, we implement the algorithm on our four-legged centaur-like robot CENTAURO and demonstrate the use of our method in assuring the stability of the robot in real-world scenarios, where the robot is traversing terrains of varying roughness.**

I. Introduction

With the increasing frequency of natural or man-made disasters, the need for mobile/legged robots to be able to function in the absence of human supervision, is of prime importance. A natural view suggests that the first step in realizing this goal involves robust environmental perception. Perceptual algorithms in indoor settings have witnessed remarkable progress in terms of reliability [1]. However, perception in unstructured outdoor scenes is still under development [2]. Outdoor environments, especially the terrain, are characterized by uncertainities, such as weather or disaster-induced changes, and variabilities that pose a major challenge to the robot perception.

Everyday experience suggests that humans are capable of reasoning about a scene using visual cues. Studies on vision-based material perception attribute this innate nature to the combined ability to recognize materials and estimate their properties, such as roughness and glossiness [3], [4]. Material recognition, as such, does not offer much knowledge by itself. However, each material encodes some implicit or a priori information, such as the slipperiness of ice, that proves to be important. The characteristics, on the other hand, assign a quantitative value to the material, which provides a high-level understanding of the environment. The complexity of the environment dictates the degree to which each perception element is required. For instance, in an urban setting, it is sufficient to identify the sidewalk for the purpose of walking. However, in a forested region, where there is no clear walking path, terrain characteristics are needed. Following this rationale, it is clear that robot perception should also involve both recognition and characterization to function in any environment. More practical examples, in support of this statement, are discussed in planetary exploration [5] and material perception [3] studies.

Robust state-of-the-art vision-based terrain recognition has been previously demonstrated through deep learning methods, such as Convolutional Neural Networks (CNNs) [6]. However, it is hard to comprehend how properties are inferred from vision. Fleming, in his study [4], suggests that the true value of the property is unknown, but the brain has a unique ability to capture the variation using straightforward appearance-based attributes, such as lighting and reflections. We use this rationale to estimate roughness along with the terrain labels. Anticipation of roughness plays an important role in path-planning, safety, and traversability analysis, such as in deciding the speed [7] and restricting the energy consumption of the robot. Although the roughness is the estimated property here, we believe that this principle can also be used for other visually noticeable properties.

Our work consists of two parts: 1) a primary CNN, which is responsible for terrain recognition and 2) a module that utilizes the low-level image features from the CNN, to capture the variation in terrain roughness (Fig. 1). This model accepts a single RGB image as an input. Generally, the three-dimensional data produced by depth-based sensors, such as RGB-D cameras (e.g., MS Kinect), laser scanners, and stereo cameras, are used to determine the geometrical properties. However, due to natural lighting conditions or material variations in an outdoor setting, the data points from such sensors may either be sparse or unreliable [8]. Moreover, reliable RGB images can also be obtained from light-weight cameras, with low energy consumption, which are the only sensor present in miniature robots such as mini-quadcopters. This makes the need for using RGB-only methods important. Moreover, using a single architecture, facilitates the sharing of parameters for two different tasks and at the same time, reduces the overall computation. We employ two important strategies, i.e., transfer learning and data augmentation, to maximize the robot's perception capability in the presence of a limited training set.

Summarizing, our main contribution in this paper is the development of a CNN-based perception system, that can simultaneously infer the terrain labels and the perceived roughness at the pixel-level from an input RGB image. We show the practical applications of our approach through experiments on the centaur-like robot CENTAURO [9]. We

[1] Department of Cognitive Robotics, TU Delft. `V.Suryamurthy@student.tudelft.nl`

[2] Department of Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego 30, 16163, Genova, Italy. {`Dimitrios.Kanoulas, Arturo.Laurenzi, Nikos.Tsagarakis`}`@iit.it`

provide the datasets, network, and code under our web-page: `https://sites.google.com/site/tsrenet`

Next, we cover the related work on segmentation and roughness estimation (Sec. II), followed by the methodology description (Sec. III). Then, we present our experimental validation of the method (Sec. IV), followed by the robotic application on the CENTAURO robot (Sec. V). We finally conclude and discuss future work (Sec. VI).

II. RELATED WORK

Early terrain segmentation methods have laid emphasis on the selection of image representation, ranging from simple colour [10] and texture [11] to Speeded Up Robust Features (SURF) [12] and feature combinations [13]. Except for the first work where each pixel is classified as vegetation or not, the others rely on patch-based classification. End-to-end learning, a key attribute of deep learning, eliminates the need for hand-picking image representations and can provide real-time segmentation. As a result, deep learning techniques, such as CNNs, have become the mainstay for image-based segmentation. CNNs use a single classifier to segment the entire image, unlike traditional approaches which require more than one detector for multi-class segmentation [14]. A brief introduction to CNNs can be found in Appendix A. In literature, terrain and material perception are used interchangeably and have the same underlying principle. Bell et al. [15] proposed a segmentation architecture consisting of a CNN, which predicts a material label for every patch extracted through a sliding window on the image, and a Conditional Random Field (CRF) that produces the segmented output based on the individual predictions. The weights of a similar patch-trained CNN, facilitated fine-tuning of a Fully Convolutional Network (FCN) [16] in Wang et al. [17]. This model operates on $4D$ light-field instead of images. Schwartz et al. [18] showed an improvement in the material recognition performance through the integration of global context. All the previous approaches depend on image patches for training. More closely related to our segmentation approach, Schilling et al. [6] trained an FCN on the cityscapes dataset [19]. The model is then adapted to the local terrain by fine-tuning the pre-trained weights on hand-labeled images. In contrast, we fine-tune our pre-trained model on a terrain specific dataset detailed in Sec. IV-A.1.

Roughness assessment has been widely used in terrain traversability analysis [20], [21]. A purely image-based approach introduced by Howard et al. [22], detects the concentration of the rock in the image and uses it as a measure of roughness. More recently in Kim et al. [23], a CNN-based architecture, taking an image and a roughness measure, almost identical to ours, as input, segments the terrain into rough, smooth and vegetated areas. Ram [24] uses a SegNet [25], a CNN with low processing time, to predict the terrain roughness whose values are discretized into four classes. Contrary to both the work which categorizes the roughness metric, our approach estimates roughness through a regression framework. Studies similar to our concept combine material or terrain recognition with

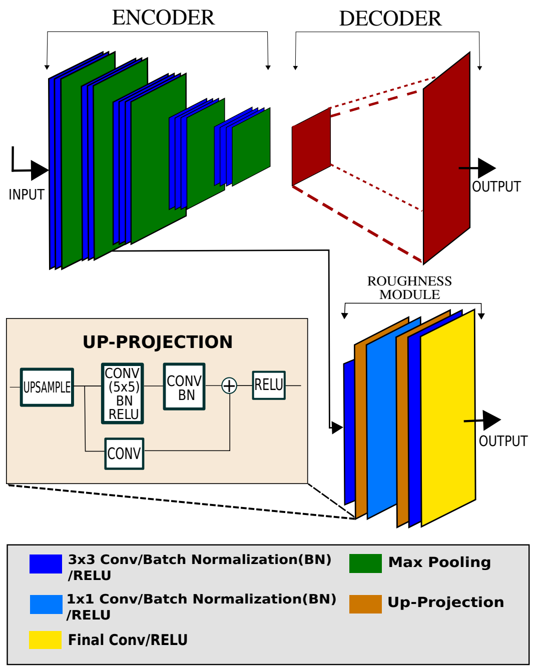


Fig. 1. The proposed architecture: the encoder-decoder system, based on SegNet, predicts the pixel-wise labels, while the roughness module regresses the perceived roughness.

friction [26] and slip estimation [27]. In the former work, the friction distribution is obtained from experiments on a humanoid robot for each terrain. The product of the priorly determined friction distribution and the material class distribution derived from a CNN, SegNet, gives the joint distribution. In the latter work, two different representations, appearance and geometry, are computed from stereo vision. Appearance features facilitate terrain recognition, which is used to learn a non-linear mapping between a geometrical property (slope) and slip. Our model, based on deep neural networks, differs from the previous work in that a single architecture learns to predict the terrain classes and regresses their roughness from an RGB image.

III. METHODOLOGY

In this section, we describe our introduced model that predicts the label and estimates the roughness of the terrain from an RGB image. The training is a two-step process: the first step involves the adaptation of a pre-trained model to terrain recognition through the process of fine-tuning and the second step involves the training of the proposed roughness module. An overview of the architecture is shown in Fig. 1.

A. Segmentation

We frame the terrain recognition problem as a semantic segmentation task predicting per-pixel labels. For this purpose, a SegNet is used. Its architecture involves an encoder that abstracts features through a pooling operation, and a decoder, which is the exact mirror image of the encoder, responsible for upsampling. Computation time and memory requirement are crucial factors that have to be considered for real-time operation on a robot, such as CENTAURO. A popularly used network, as mentioned in Sec. II, is FCN. Compared to FCN, SegNet offers a better trade-off between inference time and performance. SegNet stores the max-pooling indices for upsampling as opposed to the encoder feature maps in FCN, thereby occupying less memory. Merging the batch normalization layers after the completion of training further reduces the inference time. More details on the semantic segmentation model is provided in Appendix B.1.

Training a network from scratch may need a lot of man-hours and a large amount of training data, which as detailed in Sec. IV-A.1, may not exist. As a result, our method relies on transfer learning principles. We initialize the model with pre-trained weights derived from training the SegNet on the cityscapes dataset. Fine-tuning a pre-trained model has the advantage of better generalization and faster convergence as opposed to a freshly trained model [28]. The intuition behind this approach is that cityscapes, being an outdoor-based dataset, is closely related to the task of terrain segmentation and hence, a relevant initializer for domain adaptation (see Appendix C.1).

As ambiguity is a major problem in outdoor terrain, it is vital for the model to generalize to unforeseeable environments. We incorporate data augmentation techniques, such as random flip, and contrast/brightness change, during training, to improve the generalization of the model [29]. The images are randomly flipped both horizontally and vertically with a 0.25 probability. Moreover, with the same probability, we randomly vary the brightness and contrast. We avoid cropping, as a result of sparse annotations in the labels.

For fine-tuning, the RGB images are resized to 256×512 and a mini-batch size of 10 is used. As a result of class imbalance, the network minimizes a weighted cross-entropy loss by gradient descent with a constant learning rate of 10^{-3}, momentum of 0.9, and a weight decay of 0.0005. The weights for the loss function are computed through median frequency balancing [30](see Appendix C.3). A simple confidence measure of 0.4 thresholds the softmax output of the model. This means that any prediction with a probability less than 0.4 is set as “unlabelled”. We avoid post-processing steps, such as CRF, for computational purposes.

B. Regression

A common knowledge in deep learning is that the specificity of the feature space increases with the depth of the CNN. This leads us to our concept of using the trained bottom layers of SegNet, which correspond to primitive appearance features, to estimate the perceived roughness. Given an input image I, the bottom layers of the SegNet provides the probability distribution $P(F|I)$ from initial fine-tuning, where $F = f_1, f_2, ...f_n$ corresponds to the features derived from the convolutional layer and n is the total number of features, equal to the product of input dimension and the number of feature maps. The roughness module determines the probability distribution $P(R|F)$, where R is the roughness. Hence, at each pixel in the image, the probability distribution is given by,

$$P(R|I) = \sum_{i=1}^{n} P(R|f_i)P(f_i|I).$$

The proposed roughness module is similar to the decoder, in the sense that the Segnet layers provide sufficient feature abstraction and further pooling operation is not required. We use up-projection blocks, introduced in Fully Convolutional Residual Networks (FCRNs) [31], to increase the spatial resolution of the downsampled feature maps. This approach eliminates the need for memory-intensive fully connected layers, generally, used for deep regression and speeds up the convergence process. The roughness modules consist of three stages: a convolutional layer with 3×3 kernel, two up-projection blocks separated by a 1×1 convolution, and final convolutional layers with ReLU activation yielding the per-pixel roughness values. The 1×1 convolution performs a feature pooling operation and adds non-linearities to the model due to the presence of ReLU. Every convolution in the primary branch of the up-projection block is followed by batch-normalization. The roughness module and its setting during training is detailed in Appendix B.2.

With regards to the training, the weights of the entire segmentation model are frozen and a new branch embodying the roughness module is added to the second pooling layer of the model. Adding the module to a deeper layer than a second pooling layer did not improve the regression performance. The module is trained with the same setting as before, but at a lower learning rate of 10^{-7} and an increased batch size of 25. Apart from the conventional Euclidean loss (L_2 norm) used in regression tasks, we also experiment with berHu loss [32]. BerHu or reverse Huber loss switches between the L_1 and L_2 norm depending on a threshold c:

$$B(x) = \begin{cases} |x| & |x| \leq c \\ \dfrac{x^2 + c^2}{2c} & |x| > c \end{cases}$$

where x is the error between the prediction and the ground-truth, and $c = \frac{1}{5} max_k(|x_k|)$ is computed for every step of gradient descent for an output with k pixels. BerHu loss produces a lower final error than Euclidean loss. The reason is associated to the fact that berHu gives a higher weightage to small errors which, in general, are not accounted for by Euclidean.

IV. Experiments

In this section, we first describe the terrain and roughness datasets, followed by the visual-based results. Both datasets are available under our web-page.

A. Datasets

1) Terrain Dataset: Image datasets with pixel-wise annotations, such as ADE20K [33], have facilitated the growth of segmentation approaches. These datasets, however, are generalized to a wide variety of classes and not specific to terrain. Datasets intended for material segmentation, such as OpenSurfaces [34], are mostly indoor-based. Cityscapes,

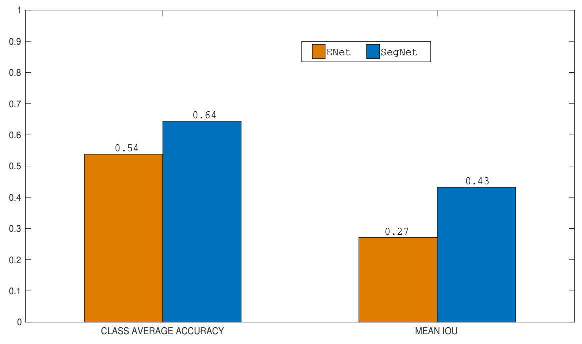


Fig. 2. Class average accuracy and mean IoU of SegNet and ENet, evaluated on the validation set.

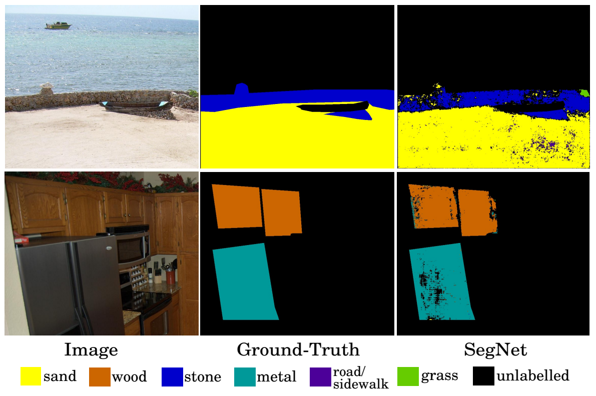


Fig. 3. Segmentation results from the validation set. **Top row:** an image from the ADE20K dataset. **Bottom row:** an image from the OpenSurfaces.

although outdoor-based, is primarily meant for autonomous driving in urban settings and hence lacks the commonly found terrain classes, such as rocks and sand. However, the classes of interest are distributed among these datasets. Our immediate thought is to extract relevant images and annotations from various publicly available segmentation datasets and combine them. Resorting to this approach, allows us to harness the already existing datasets, thereby eliminating the need for hand-labelled annotations, which is a time-consuming process. A short discussion on the datasets considered for segmentation is provided in Appendix D.1.

We construct a dataset with six most frequently occurring terrain classes (sand, stone, wood, metal, road/sidewalk, grass). The similarly appearing road and sidewalk are combined into a single class for unambiguous prediction. The dataset is built such that it shows maximum variability in the scenes and minimal imbalance between the terrain classes. A large imbalance causes conservative biases to the predictions, even when using weighted loss function. As a consequence, we first select a subset of images, both for training and validation, from ADE20K as it is a huge dataset consisting of images taken in various environments. However, ADE20K has sparse annotations with regards to wood and metal surfaces. Therefore, we use images from OpenSurfaces to accommodate the sparsity. In all, the dataset contains 1380 images, of which 65% is allocated for training, while the remaining 35% for validation. We test the model on images collected from mobile RGB cameras and the Kinect v1 RGB-D sensor, mounted on the CENTAURO robot.

2) Roughness Dataset: Roughness is a visually noticeable, geometrical property that describes the irregularities in the material surface. For training the CNN to regress roughness, a dataset is required. Numerous methods [7], [8], [23], [21] have been used to compute the roughness measures. We use a simple approach suggested by Gennery [20], wherein the geometric residual of the plane fit on 3D points of the surface, using a least squares approach, provides the roughness measure. The 3D depth data corresponding to terrain patches is provided by GeoMat [35]. The dense depth data in the dataset is computed using Structure from Motion (SfM), followed by interpolation. The dataset contains patches of four different sizes (100×100, 200×200, 400×400, and 800×800) cropped from a high resolution image and resized to a 100×100 patch. These high-resolution images provide surface roughness at small local regions, which is of less interest to the robot. We look for a more global roughness in the scale of robot parts, such as the wheel/footpad surface that is in contact with the terrain. Hence, the extracted dataset involves the downsampled images (400×400 and 800×800) only. The training and validation set has 1375 and 880 images, respectively.

The depth images are initially transformed into point clouds, which are further converted to roughness values using the Point Cloud Library (PCL) [36] (see Appendix D.2). From every point in the point cloud, the nearest neighbours are computed using K-Nearest Neighbours. We find the value of K$=9$ suitable for the 100×100 downsampled patches. The plane parameters are estimated for the given nearest neighbour patch. Given the best fit plane equation:

$$ax + by + cz + d = 0$$

The roughness at each point (x_i, y_i, z_i) is computed by,

$$r_i = \frac{|-d - ax_i - by_i - cz_i|}{\sqrt{a^2 + b^2 + c^2}}$$

where i is the index of the point.

B. Experimental Results

In this section, we discuss the results on the performance of segmentation and roughness estimation. Our implementation is in Caffe [37]. The pre-trained weights for fine-tuning are available in the caffe model zoo.

1) Segmentation: To evaluate the segmentation performance, we use two metrics; class average accuracy and mean Intersection over Union (IoU), introduced in the Pascal VOC12 challenge [38]. As inference time is a key factor for real-time operation on the robot, we consider ENet [39] along with SegNet for our task. ENet is reported to have a really low test time. Fig. 2 shows the evaluation of the two architectures on the validation set, using the class average accuracy and mean IoU.

It can be seen that SegNet has a higher accuracy than ENet on our dataset. After merging the batch normalization

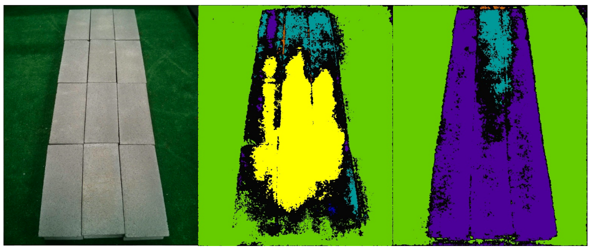


Fig. 4. Effect of data augmentation techniques. **From left to right:** an image from an RGB camera, prediction without, and with brightness and contrast adjustments.

and dropout (in ENet), the difference in testing time of ENet when compared to SegNet for a 256×512 image (10.25ms vs 18.54ms) is not significantly large. Hence, we find the SegNet suitable for our purpose. Further, we observe an additional improvement in the prediction when the first convolutional layer of the model is frozen and the training is restricted to 50 epochs. We attribute the reason to the relatively small size of the dataset which can result in overfitting. Examples of the segmentation results of SegNet on the validation set is shown in Fig. 3.

The data augmentation techniques counter the biases and ambiguous labelling in the dataset. For instance, the labels corresponding to the class "path" in the ADE20K dataset contain both sand and road. Minor adjustments in random brightness and contrast mitigate this confusion resulting from ambiguous labels, as seen in Fig 4, and therefore, increase the generalizability and predictive power of the model.

2) Roughness: We evaluate the per-pixel regression of roughness using the Root Mean Square Error (RMSE) and Root Mean Square Log Error (RMSLE). Table I compares the final error of our model to those reported in the only related work [24] that has attempted to estimate roughness in a regression framework. This approach uses image classification CNNs (VGG16, Cifar10) and a custom designed model (terrainNet) to regress a roughness measure, whose ground-truth is based on acceleration data from IMU. However, as a result of large mean squared errors, the problem is converted to a segmentation problem by dividing the range of roughness values into four classes.

TABLE I
COMPARISON OF THE PROPOSED METHOD WITH STATE-OF-THE-ART APPROACH

Architecture	Optimizer	Loss	rmse	rmsle
TerrainNet	RMSprop	Euclidean	6.4219	
cifar10	Adadelta	Euclidean	5.9741	
Ours	SGD	Euclidean	0.4350	0.1123
		Berhu	**0.4335**	**0.1119**

Despite the small difference in loss, it is observed that the overall estimation of roughness is significantly improved when using the berHu loss function. Qualitatively, our approach successfully captures the variation in roughness in spite of the noise resulting from inaccuracies in SfM computed ground-truth depth data. However, the magnitude of roughness is downscaled in the estimated output (see Fig. 5). The roughness range of $0 - 3.7$ cm in the ground-truth is reduced to $0.05 - 1.4$ cm in the regressed output.

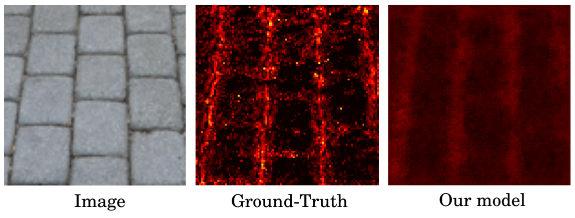


Fig. 5. Examples of roughness estimation on the transformed GeoMat dataset. Both heatmaps (ground truth and our model's) are scaled equally.

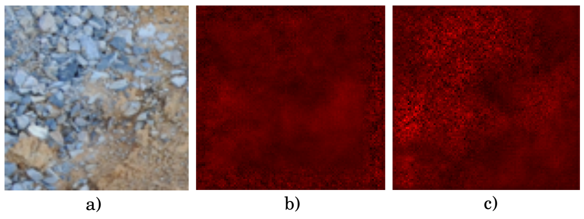


Fig. 6. a) The original image, b) regressed output from the freshly trained model, and c) regressed output derived using the appearance-based features.

The most probable explanation is that the downscaling is caused by a bias in our dataset, where several instances of the surfaces are flat or nearly flat. The gradient descent driven training tends to conservatively estimate values of low magnitude. This becomes apparent when the same model is trained from scratch in the absence of image features. In fact, using the appearance-based features lessen this bias, as seen in Fig. 6.

V. ROBOTICS APPLICATION

In this section, we briefly discuss the need for segmentation and roughness on legged robots. Roughness is one of the characteristics that directly provide information on the terrain traversability. For legged locomotion, the terrain roughness is all the more important as a number of parameters, such as the Center-of-Mass (CoM) location and speed, have to be optimized to ensure fail-safe operations. However, roughness is not very much applicable to non-rigid terrain, for instance, grass and sand, which are mostly deformable. This can also be seen in [23] where the authors consider vegetation as a separate class independent of rough and smooth areas. In such situations, the implicit knowledge associated with a terrain class enables better decision making in the robot. These decisions can be in the form of risk measures when a particular terrain class is identified. We run a series of tests on the wheeled quadrupedal robot CENTAURO (Fig. 7) to study the parameters that influence the traversability of the terrain with varying roughness.

A. *The CENTAURO Robot*

CENTAURO is a centaur-like 42-DoF robot that weighs around 90 kilograms. Its upper body is human-like, with two

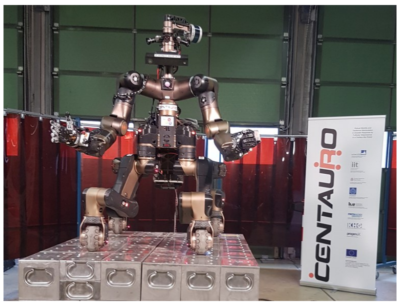


Fig. 7. The wheeled quadrupedal centaur-like robot CENTAURO.

7-DoF arms and a head equipped with a spinning 3D LiDAR scanner, an RGB-D Kinect v1 sensor, and three monocular RGB cameras. Its lower body consists of four 6-DoF legs, each of which has a wheel at the end.

CENTAURO is controlled through the real-time (RT) robotic framework *XBotCore* [40]; on top of it, the Cartesian control of CENTAURO is made possible by the *CartesI/O* framework[1], which provides a set of Robot Operating System (ROS) topics, services, and actions that are used to send Cartesian references; for our experiments, we employed a custom module from CartesI/O, which is tailored to the hybrid wheeled-legged locomotion of the CENTAURO robot. Such a module allows for full 6D pose control of the robot CoM through appropriate wheel steering and spinning. Moreover, the support polygon shape can be dynamically changed.

B. Online Segmentation and Roughness Estimation

First, we demonstrate the real-time terrain segmentation and roughness estimation on CENTAURO. For the experiment, the RGB image input from the Kinect v1 is constantly fed into the model at 30 frames per second, while the roughness of the surface is manually varied. Fig. 8 shows some examples of the segmented output and estimated roughness from the experiment. Despite implementing techniques to improve the generalizability, we still notice some segmentation failures in the last two rows in the figure, where certain areas in large stones are confused as road/sidewalk which might be a result of similar appearances. Given an image of 256×512 dimensions, the test time of the model is 30.59ms. This is presented for equal comparison with the results shown in Sec. IV-B. However, it is sufficient for the robot to comprehend the closest observable area of suitable height and width whose value is slightly more than the maximum wheel-to-wheel distance. As a result, the region of interest is a 192×384 crop from the bottom portion of the original image, which is taken as the input in this experiment. The regressed roughness seems to generalize well to inputs of different dimension. This underlies the flexibility of the model with regards to the input image size.

It is observed that both segmentation and roughness outputs are affected by motion blur. Inference in the presence of motion blur is an area of future research.

[1]`github.com/ADVRHumanoids/CartesianInterface`

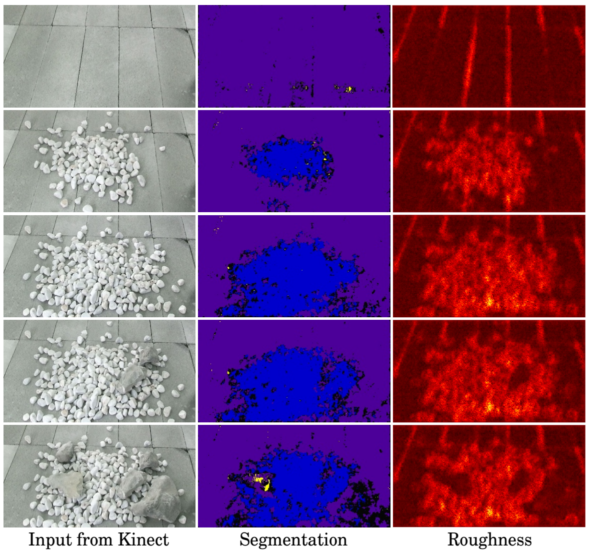


Fig. 8. Example results of segmentation and roughness estimation from the experiment. The heatmap scale is reduced to half for better visualization.

C. Influence of CoM Height and Support Polygon

A rough terrain may decrease the stability of the robot. When the perturbations from the surface are large in magnitude, there are instances where the number of contact points is less than three. This makes the robot unstable and in extreme conditions, it may result in a fall. Enlarging the support polygon or decreasing the height of CoM can reduce any possibility of a fall in a four-legged robot. In the case of CENTAURO, due to its kinematics, decreasing the CoM height, automatically increases the support polygon. We have experimentally validated the danger of rolling with straight legs over some rough terrain, using CENTAURO in the setting shown in Fig. 9, which resulted in very unstable phases.

However, increasing the support polygon comes at an expense of higher torques on the joint motors, and thus higher energy consumption. The smallest support polygon is achieved when the robot legs are straight with minimal joint torque. Hence, an optimal height/support-polygon has to be computed, depending on the roughness of the terrain. We demonstrate a simple experiment in an artificially constructed environment, wherein the robot alters the height of its CoM based on a terrain roughness threshold, as shown in Fig. 9. In this paper, with the Cartesian Interface (Sec. V-A), we allow the robot to change its CoM height/support-polygon, from fully stretched legs, where the robot's CoM is at 0.95m from the ground with a small support polygon of 0.36m^2, to bent legs, where the robot's CoM is at 0.75m from the ground with a larger support polygon of 0.63m^2.

A roughness output of 0.3cm from the model is considered as the threshold [2]. This can be related to how human perceive

[2]Notice, that the output of the model is directly taken for the roughness threshold. As our model captures the variation in roughness, a post-processing step, such as upscaling to the original range, is possible.

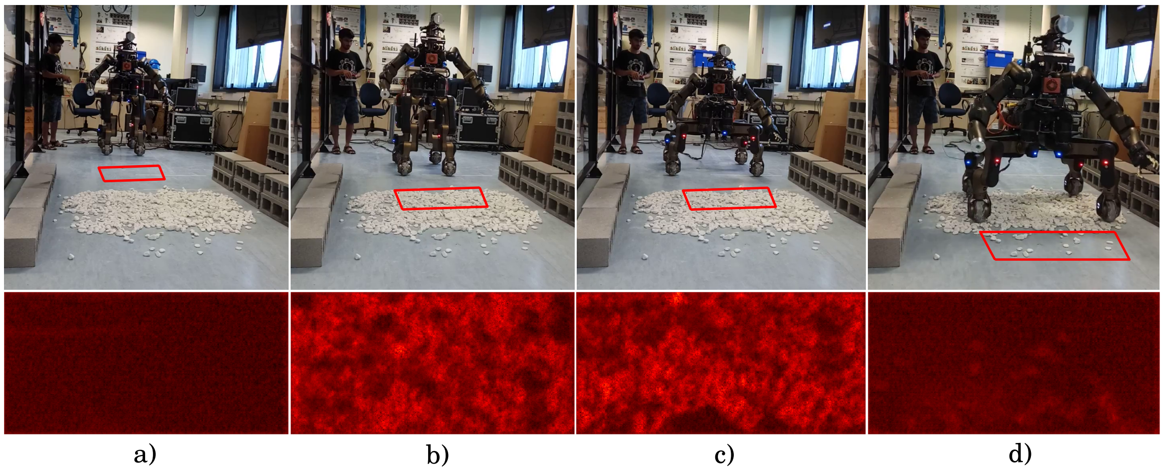


Fig. 9. Different stages of the experiment, along with the estimated roughness. a) Upright position (i.e., stretched) of CENTAURO on the flat surface, b) rough terrain detection from our network in front of CENTAURO front wheels, c) CENTAURO lowers its CoM (i.e., bent), and d) traversing the rough terrain.

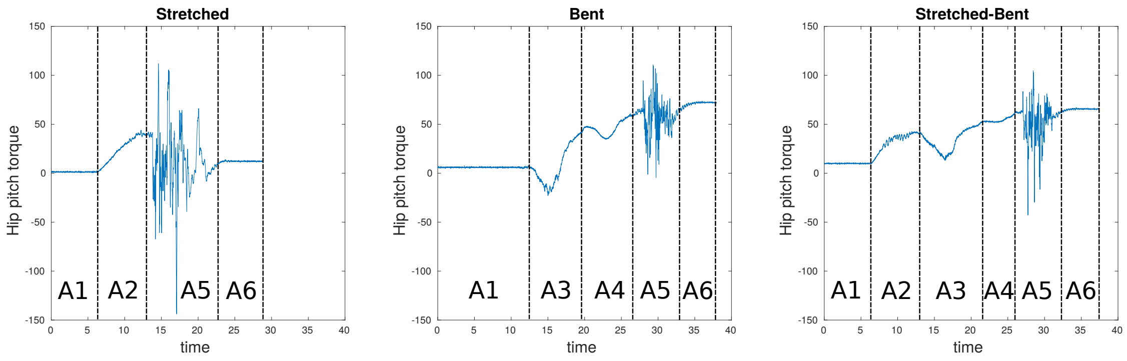


Fig. 10. The torques produced at the hip pitch joints, during three experiments: 1) when CENTAURO traverses the terrain stretched up (i.e., stretched: upright stretched configuration), 2) when CENTAURO traverses the terrain lowered down (i.e., bent: low CoM configuration), and 3) when CENTAURO traverses the flat terrain stretched up (i.e., stretched) and lowers down its CoM (i.e., bent), after detecting a rough upcoming terrain. Each sub-figure splits into the following areas: [**A1**] static upright configuration; [**A2**] acceleration with upright configuration; [**A3**] transition from upright to low; [**A4**] acceleration with low configuration; [**A5**] traversing rough terrain in corresponding configurations; and [**A6**] deceleration to rest.

roughness. Although the exact magnitude is not known, the variation or relative roughness is sufficient to interact with the environment. When the majority of the pixels in front of the robot front-wheels have a roughness value above this threshold, the robot lowers its body by increasing the support polygon. For a given speed, the robot is allowed to roll for a certain duration after the transition from rough to flat is detected. This ensures that the robot has cleared the rough terrain. The plot corresponding to one of the four hip joints is shown in Fig. 10.

For the given setting, three different experiments with high CoM (stretched), low CoM (bent) and adaptable CoM (stretched-bent) are shown in the above plot. We consider hip joints alone because knee and ankle joints are nearly in line with the reaction force from the wheels and hence does not have a significant torque variation. Looking at the first experiment, where the leg joints of the robot are almost fully stretched, the magnitude of the torque is nearly zero. However, the amplitude of torque fluctuations corresponding to the vibrations on rough terrain is quite large, which indicates that the robot is unstable (changes in the torque sign between negative and positive values, indicate loss of contact for the particular leg). In the second experiment, when the CoM of the robot is low with a larger support polygon, the energy consumption is more with the permanent increase in the magnitude of torque (see [A6] for a clear picture), but the fluctuations on the rough terrain are comparatively smaller and loss of contact happens rarely. From these observations, it is clear that the robot has to lower its CoM on rough terrain and remain upright for flat or nearly flat surfaces. This is required for minimum energy consumption and maximum stability in CENTAURO, which is demonstrated through the final experiment.

VI. Conclusions and Future work

In this paper, we introduced a method that predicts the per-pixel terrain labels and regresses roughness with CNN. Our approach involves training the SegNet on a custom-built dataset suited for terrain segmentation, and a roughness module that uses the basic image features from the initial layers of SegNet. We have showed the real-time implementation of our algorithm on the CENTAURO robot, along with a simple experiment to demonstrate the use of roughness in maintaining the stability of the robot.

In the future, we aim to test the roughness module with more segmentation networks that have higher accuracy and low computation requirements. Currently, the images of varying dimensions, from the terrain segmentation dataset, are reduced to a constant size. A scope for increase in segmentation performance lies in adaptive resizing of images such that the aspect ratio is conserved. Further, we plan on improving the roughness dataset, such that it exhibits more variability. For the robotic applications, a heuristic approach was adopted to show the relationship between the CoM location and roughness. We plan on developing a mathematical formulation involving more parameters, such as speed, in order to implement a path-planner for search-and-rescue operations. Finally, we intend to combine our introduced vision-based approach, with other approaches, e.g. force/torque-based [41], to allow more autonomous control and planning strategies during robot locomotion or navigation.

Acknowledgment

This work is supported by the CENTAURO (grant agreements no 644839) EU project. The Titan Xp used for this research was donated by the NVIDIA Corporation.

The authors would like to thank Prof. Martijn Wisse for his useful comments and recommendations, as well as Joseph DeGol for his help on the GeoMat dataset.

References

[1] P. Papadakis, “Terrain Traversability Analysis Methods for Unmanned Ground Vehicles: A Survey,” *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1373–1385, 2013.

[2] A. Singhal, “Issues in Autonomous Mobile Robot Navigation,” Tech. Rep., 1997.

[3] E. H. Adelson, “On Seeing Stuff: the Perception of Materials by Humans and Machines,” vol. 4299, 2001, pp. 1–12.

[4] R. W. Fleming, “Visual Perception of Materials and their Properties,” *Vision Research*, vol. 94, pp. 62–75, 2014.

[5] E. Tunstel and A. Howard, “Sensing and Perception Challenges of Planetary Surface Robotics,” in *IEEE Sensors*, vol. 2, 2002, pp. 1696–1701.

[6] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt, “Geometric and Visual Terrain Classification for Autonomous Mobile Navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2678–2684.

[7] M. Castelnovi, R. Arkin, and T. R. Collins, “Reactive Speed Control System Based on Terrain Roughness Detection,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 891–896.

[8] D. Stavens and S. Thrun, “A Self-supervised Terrain Roughness Estimator for Off-road Autonomous Driving,” in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, ser. UAI’06, 2006, pp. 469–476.

[9] L. Baccelliere, N. Kashiri, L. Muratore, A. Laurenzi, M. Kamedula, A. Margan, S. Cordasco, J. Malzahn, and N. G. Tsagarakis, “Development of a Human Size and Strength Compliant Bi-Manual Platform for Realistic Heavy Manipulation Tasks,” in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 5594–5601.

[10] I. L. Davis, A. Kelly, A. Stentz, and L. Matthies, “Terrain Typing for Real Robots,” in *Intelligent Vehicles Symposium.*, 1995, pp. 400–405.

[11] M. Marra, R. Dunlay, and D. Mathis, “Terrain Classification Using Texture For The ALV,” *Cambridge Symposium on Advances in Intelligent Robotics Systems (SPIE)*, vol. 1007, pp. 1–8, 1989.

[12] P. Filitchkin and K. Byl, “Feature-based Terrain Classification for LittleDog,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 1387–1392.

[13] A. Angelova, L. Matthies, D. Helmick, and P. Perona, “Fast terrain classification using variable-length representation for autonomous navigation,” in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[14] E. Romera, L. M. Bergasa, and R. Arroyo, “Can we Unify Monocular Detectors for Autonomous Driving by using the Pixel-Wise Semantic Segmentation of CNNs?” *CoRR*, vol. abs/1607.00971, 2016.

[15] S. Bell, P. Upchurch, N. Snavely, and K. Bala, “Material Recognition in the Wild with the Materials in Context Database,” *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[16] J. Long, E. Shelhamer, and T. Darrell, “Fully Convolutional Networks for Semantic Segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[17] T.-C. Wang, J.-Y. Zhu, H. Ebi, M. K. Chandraker, A. A. Efros, and R. Ramamoorthi, “A 4D Light-Field Dataset and CNN Architectures for Material Recognition,” in *European Conference on Computer Vision (ECCV)*, 2016.

[18] G. Schwartz and K. Nishino, “Material Recognition from Local Appearance in Global Context,” *CoRR*, vol. abs/1611.09394, 2016. [Online]. Available: http://arxiv.org/abs/1611.09394

[19] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[20] D. B. Gennery, “Traversability Analysis and Path Planning for a Planetary Rover,” *Autonomous Robots*, vol. 6, no. 2, pp. 131–146, 1999.

[21] S. B. Goldberg, M. W. Maimone, and L. Matthies, “Stereo Vision and Rover Navigation Software for Planetary Exploration,” in *IEEE Aerospace Conference*, vol. 5, 2002, pp. 2025–2036.

[22] A. Howard and H. Seraji, “Vision-based Terrain Characterization and Traversability Assessment,” *Journal of Robotic Systems*, vol. 18, no. 10, pp. 577–587, 2001.

[23] D.-K. Kim, D. Maturana, M. Uenoyama, and S. Scherer, “Season-Invariant Semantic Segmentation with A Deep Multimodal Network,” in *Field and Service Robotics*, September 2017.

[24] S. Ram, “Semantic segmentation for terrain roughness estimation using data autolabeled with a custom roughness metric,” Master’s thesis, Carnegie Mellon University, Pittsburgh, PA, July 2018.

[25] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2017.

[26] M. Brandao, Y. M. Shiguematsu, K. Hashimoto, and A. Takanishi, “Material Recognition CNNs and Hierarchical Planning for Biped Robot Locomotion on Slippery Terrain,” in *16th IEEE-RAS International Conference on Humanoid Robots*, Nov 2016, pp. 81–88.

[27] A. Anelia, M. Larry, H. Daniel, and P. Pietro, “Learning and Prediction of Slip from Visual Information,” *Journal of Field Robotics*, vol. 24, no. 3, pp. 205–231.

[28] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How Transferable Are Features in Deep Neural Networks?” in *27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS’14. MIT Press, 2014, pp. 3320–3328.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS’12. USA: Curran Associates Inc., 2012, pp. 1097–1105.

[30] D. Eigen and R. Fergus, “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture,” in *IEEE International Conference on Computer Vision (ICCV)*. Washington, DC, USA: IEEE Computer Society, 2015, pp. 2650–2658.

[31] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, "Deeper Depth Prediction with Fully Convolutional Residual Networks," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 239–248.

[32] A. B. Owen, "A Robust Hybrid of Lasso and Ridge Regression," Tech. Rep., 2006.

[33] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic Understanding of Scenes Through the ADE20K Dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[34] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "OpenSurfaces: A Richly Annotated Catalog of Surface Appearance," *ACM Trans. on Graphics (SIGGRAPH)*, vol. 32, no. 4, 2013.

[35] J. DeGol, M. Golparvar-Fard, and D. Hoiem, "Geometry-Informed Material Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[36] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[38] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int. Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[39] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation," *CoRR*, vol. abs/1606.02147, 2016.

[40] L. Muratore, A. Laurenzi, E. M. Hoffman, A. Rocchi, D. G. Caldwell, and N. G. Tsagarakis, "Xbotcore: A Real-Time Cross-Robot Software Platform," in *IEEE Int. Conference on Robotic Computing (IRC)*, 2017.

[41] K. Walas, D. Kanoulas, and P. Kryczka, "Terrain Classification and Locomotion Parameters Adaptation for Humanoid Robots using Force/Torque Sensing," in *IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, 2016, pp. 133–140.

A

Introduction to Convolutional Neural Networks

Among the various types of neural networks, Convolutional Neural Networks (CNNs) stand out for its application to computer vision, especially with tasks involving images. The reason is two-fold:

1. High dimensional images are effectively handled through minimal number of parameters, thereby making it feasible to train models on a graphics processing unit (GPU).

2. As opposed to the conventional neural networks, CNNs preserve the structure of the image in the sense that the mathematical operations for feature extraction are two dimensional, and the learnable feature maps are aggregated along the third dimension similar to that of R,G,B colour channels in an image (see Fig. A.1).

Additionally, the availability of huge image datasets and the advantages of end-to-end learning have given the data-driven CNNs an edge over the state-of-the-art methods.

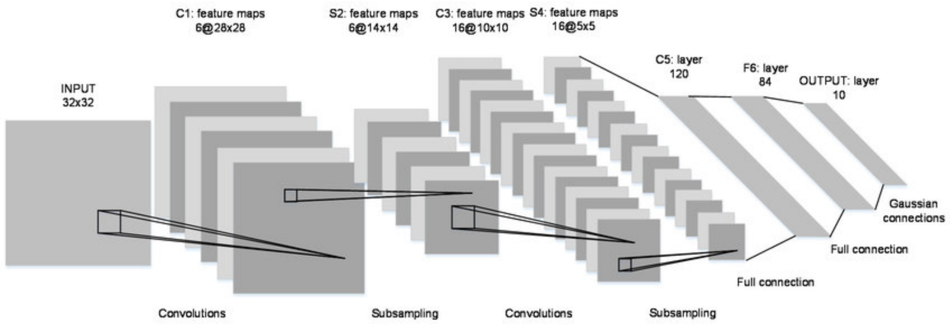


Figure A.1: An example CNN architecture. It depicts the basic operations, such as convolutions and pooling (subsampling), involved in every CNN. Note the accumulated feature maps along the layer's depth. The fully connected layer (full connection) is a special case of convolution wherein the two-dimensional feature map is reduced to a single element. The architecture shown here is LeNet [1], the first proposed CNN architecture.

A CNN takes an image of dimensions $w \times h \times 3$ as input, extracts a set of representations, and produces the necessary output using a classifier at the end according to the task at hand. The representation space is learnt during the training of a series of layers that form the core of a CNN. Most of our architecture is composed of four commonly found layers; Convolutional layer, Pooling layer, Rectified Linear Units (ReLU), and Batch Normalization. These are discussed below,

- A unique feature of CNNs is the use of learnable convolution kernels (similar to that of convolutional filters used in image processing) that serve as weights for the model. Each kernel performs filter operation, mostly elementwise addition, on the image. These kernel operations are carried out by the

Convolutional layer. The kernel size (known as receptive field) are, in general, small in size, usually 3×3 or 5×5. As the receptive field is much smaller than the image size, the parameter count is low. To further reduce the number of parameters involved, a single kernel, corresponding to an output channel, is shared/slid across all the locations of the input. For N output feature maps, the convolutional layer consists of N learnable kernels. There are special cases of receptive fields such as 1×1 and $m \times n$, where m is the width and n is the height of the image. The former provides an efficient way of increasing or decreasing the depth (number of feature maps) of a convolutional layer. In our case, it is used for feature abstraction. The latter, which is shown as **full connection (fully connected layer)** in Fig. A.1, convolves with the entire image to produce a single value output. This is an important element for image classification problems, where the final output is a $1 \times 1 \times k$ and k refers to the number of classes.

- **Pooling layer** is responsible for reducing the spatial dimension of the input through the process of downsampling. A commonly used pooling operation is max-pooling wherein a patch, say 2×2, in the input is reduced to a single element corresponding to the maximum value of the patch (See Fig. A.2). The primary role of pooling is to lessen the computational costs by reducing the number of mathematical operations in the model. A similar but opposite function is carried out by the **Unpooling/Deconvolutional layer**.

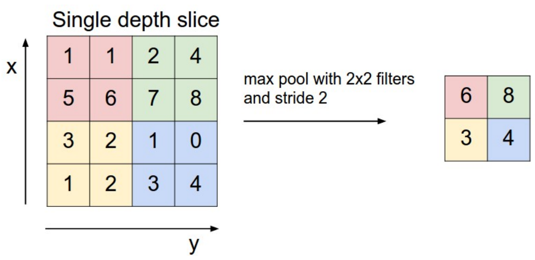


Figure A.2: Maxpooling [2]

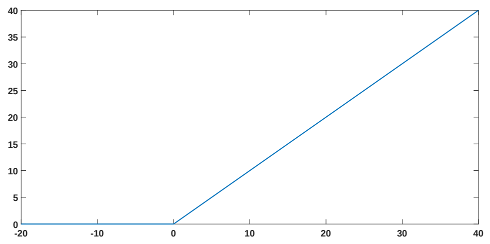
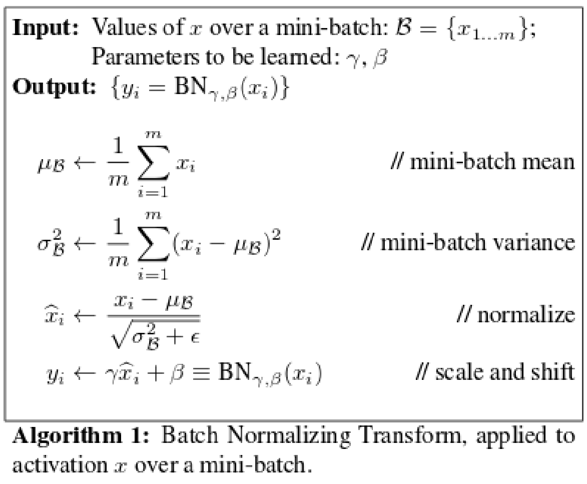


Figure A.3: ReLU

- **ReLU** [3] is a popular activation function that has been proved to offer better performance than the standard sigmoid or tanh functions. ReLU has a hard non-linearity at zero which promotes sparse and robust representations in the model. Contradictory to sigmoid or tanh, ReLUs do not constrain the output to a small range of values thereby reducing the possibility of vanishing gradients. Moreover, the mathematical operation of ReLU, $max(0, x)$ (see Fig. A.3), and its gradient are simple to compute.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Figure A.4: Batch Normalization algorithm

- **Stochastic gradient descent** (SGD), a popular optimizer for training CNNs, is the iterative approximation of the gradient descent algorithm based on the gradients computed over a small set of inputs (mini-batch). As each mini-batch has a distribution of its own, the input distribution to every layer varies, thereby destabilizing the training and slowing down the convergence. **Batch Normalization** [4] (BN) alleviates this problem by normalizing the input distribution to zero mean and unit variance for a given mini-batch. Normalization ensures distributions with limited inter-batch variations which speeds up the convergence. However, the representations learnt by the layers are different from that of the un-normalized input. So, BN incorporates two parameters (a scale and a shift) that learn, during training, to restore the representations corresponding to the original activations. BN's algorithm, as proposed in the original paper [4], is shown in Fig. A.4.

As the values of these two parameters are constant during testing, the BN can be merged with the convolution layer by making a permanent scale and shift of the normalized activations. Merging the batch normalization reduces the inference time due to the elimination of normalization and scale/shift operation. Although the improvement is marginal, it makes a difference for practical applications. Table A.1 shows the observed drop in inference time when BNs are merged in our approach.

SegNet	Before merge	After merge
time (ms)	23.4988	18.54

Table A.1: Inference time before and after merging BN

An in-depth discussion on CNNs can be found in [2].

B
Architecture Design

Our CNN- based model predicts the terrain class and its roughness for each pixel. Here, we discuss and reason out the design elements of our model in two separate sections, one for each sub-task, i.e., terrain recognition and roughness estimation.

B.1. Terrain Recognition

B.1.1. Semantic segmentation over Classification

Recognition can be considered either as a classification or a semantic segmentation task. The former predicts a single label for the entire image while the latter determines the label for each and every pixel in the image. For our problem, pixel-wise understanding is of importance as real-world scenes will most likely comprise a mixture of terrains that can vary from one pixel to another. As a result, we frame the terrain recognition as a semantic segmentation problem.

B.1.2. Selection of Segmentation model

Segmentation demands the output to have the same dimensions as the input. Popular CNN architectures, such as LeNet [1], VGG-16 [5], and ResNet [6], are meant for image classification and cannot be directly used. An alternative that has been explored is patch-based classification, where the classification network takes in a sliding patch as input. However, this approach requires further processing in order to combine the individual patch outputs into a final segmentation map. Fully Convolutional Networks (FCN) [7], for the first time, introduces an end-to-end trainable architecture that produces pixel-wise segmentation maps. FCN uses the same architecture as VGG-16 and replaces the fully connected layers with convolutional layers. Observe the difference between the two architectures from Figs. B.1 and B.2. Furthermore, bilinear upsampling with deconvolutional layers and pooled feature maps are used to generate the segmented output with dimensions equal to that of input.

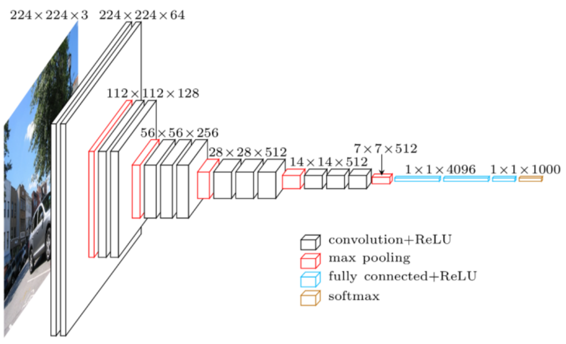


Figure B.1: VGG-16

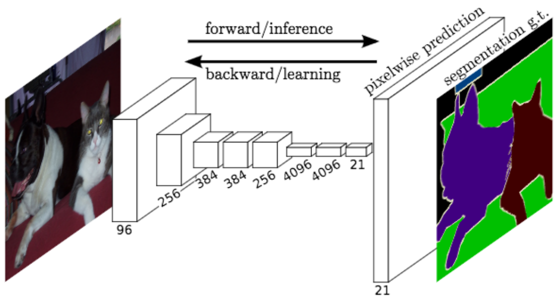


Figure B.2: FCN

Real-time inference is an important requirement for any perception system developed for robots. In our case, the Kinect v1 RGB-D sensor mounted on CENTAURO runs at 30 frames per second (fps) which poses a constraint in choosing the segmentation model. In FCN, additional floating point operations for concatenation at every upsampling stage and the large number of kernels in the replaced convolutional layers (2×4096), makes FCN unsuitable for real-time inference.

A similar architecture, SegNet can be considered as a computationally efficient counterpart of FCN. Although the encoder of SegNet is same as that of FCN, the decoder is different as it relies on pooling indices for upsampling (see Fig. B.3). Pooling index refers to the location of the maximum element in case of a max-pooling operation. As the common pooling is carried out on a 2×2 patch, the index can be stored in the form of 2 bits for every patch. This index is merely used as a pointer to the unpooled patch during upsampling and no computational costs are incurred in the process. The overall inference time and memory usage are low for SegNet, making it appropriate for the robot.

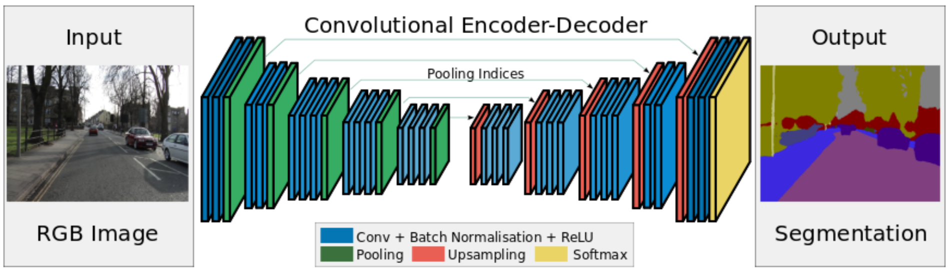


Figure B.3: SegNet

Besides SegNet, another architecture that is interesting in a practical perspective is ENet [8]. This network is specifically designed to have high computational efficiency. Various design choices, that improve the inference speed, are incorporated in the model based on the analysis of previous architectures. For example, early downsampling, by a factor of 4, reduces the parameter count drastically and use of asymmetric kernels, such as 5×5 convolution split into 1×5 and 5×1, retain the same accuracy as symmetric kernels but at a reduced cost. Table B.1 shows the inference time and the fps at which the model can operate.

Model	FCN-8	SegNet	ENet
time (ms)	71.4011	18.54	**10.25**
fps	14	54.79	**97.56**

Table B.1: Inference time

With regards to the performance, the mean Intersection of Union (IoU), an accuracy measure for semantic segmentation, benchmarked on well-known segmentation datasets is observed to be nearly the same for all these architectures, as seen in Table B.2.

Model	CamVid [9]	SUN RGB-D [10]	Cityscapes [11]
FCN-8	0.5080	0.2633	**0.6530**
SegNet	**0.5718**	**0.2746**	0.5610
ENet	0.5130	0.1970	0.5830

Table B.2: Mean IoU benchmarked on popular segmentation datasets

Given the accuracy and inference time, we consider SegNet and ENet for our problem. Both the models along with the pre-trained weights are open-sourced in Caffe. More accurate models, summarized in [12], are excluded from our analysis as it is not feasible to implement them on the robot.

B.1.3. CHOICE OF HYPER-PARAMETERS

The hyper-parameters are defined in a file called solver.prototxt in Caffe. The optimization for our approach involves minimizing the categorical cross-entropy loss using SGD. The cross-entropy loss for a pixel is given by,

$$L = -\sum_{i=1}^{k} y \log(\hat{y})$$

where y is the ground-truth label, $\hat{y}$ is the predicted probability of the pixel belonging to a given class, and k is the number of classes. For categorical cross entropy, as a result of one-hot encoding, the loss reduces to a single element corresponding to the positive class label. For example, in a six-class problem, if a label has a value of 5, the loss is equal to just the log likelihood of class 5 because the ground-truth label (y) for class 5 is one and the others are zero as per one-hot encoding. We use a softmax classifier as the last layer, the output of which is the probability for each class. This loss is summed up for every pixel in the image and averaged over the mini-batch.

Determining the ideal learning rate for this minimization problem is a challenge. As recommended in Caffe, the initial training is carried out with a learning rate of 0.01. The mean IoU has a low peak value at this learning rate. This can be due to large weight updates that prevent convergence to a minimum. To ascertain this hypothesis, we reduce the learning rate by a factor of 10 at which the loss decreases even more and converges to a high value. As a further drop in learning rate results in slow convergence, we use the learning rate of 0.001 for our approach. The mean IoU plot at two different learning rates is shown in Fig. B.4.

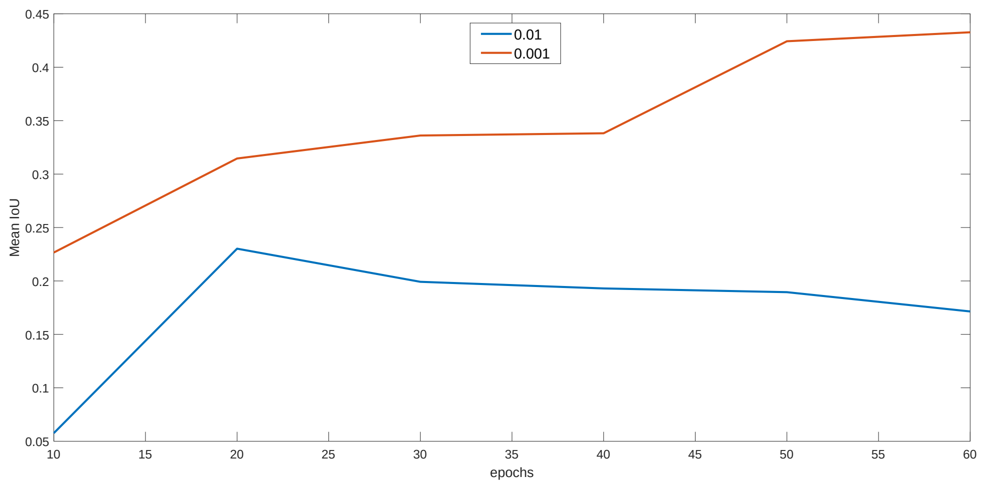


Figure B.4: Mean IoU at learning rate of 0.01 and 0.001

For the given learning rate, the standard setting, recommended in [13], with momentum of 0.9 and weight decay of 0.005 is used. Momentum accelerates the gradient descent if it keeps moving in the right direction, thereby improving the convergence rate. The formulation, as defined in Caffe, is

$$V_{t+1} = \mu V_t - \alpha \, grad(L)$$

$$W_{t+1} = W_t + V_{t+1}$$

where μ is the momentum, α is the learning rate, and $grad(L)$ is the gradient of the loss.

Weight decay penalizes large weights and has an important role especially when using ReLU activations. As the ReLUs do not have a saturation region, the magnitude of activations is kept low by the weight decay hyper-parameter.

Training with these hyperparameters, SegNet shows a better performance than ENet on our validation set, as seen from Fig. 2. and Table B.3.

Model	Sand	Wood	Stone	Metal	Road	Grass
SegNet	0.4203	0.6218	0.5146	0.4369	0.2165	0.5966
ENet	0.3627	0.5903	0.2528	0.2394	0.1969	0.5667

Table B.3: Per-class IoU of SegNet and ENet on our dataset

B.2. TERRAIN ROUGHNESS ESTIMATION

Unlike semantic segmentation, the terrain roughness parameter belongs to the continuous space and a regression framework is required to estimate those values. The introduced roughness module regresses at each pixel in the image. We analyze the roughness module in the next section.

B.2.1. UNPOOLING OVER FULLY CONNECTED LAYER

A common approach to deep regression problems, such as pose estimation, is to have a fully connected layer at the end to estimate the continuous values [14]. In such problems, the number of regressed values is mostly a single digit number and hence feasible. However, this approach is not possible for high-dimensional images since the number of parameters involved is very high. We try regressing a 100×100 image (weights = input feature map dimension$\times$10000) using a fully-connected layer but it fails as the size of the model exceeds the 12 GB memory capacity of the GPU. Successful per-pixel regression methods, especially the widely studied depth estimation task [15, 16], employ some sort of upsampling to reduce the number of weights in the model. As a result, our approach uses the deconvolutional layer of SegNet for upsampling.

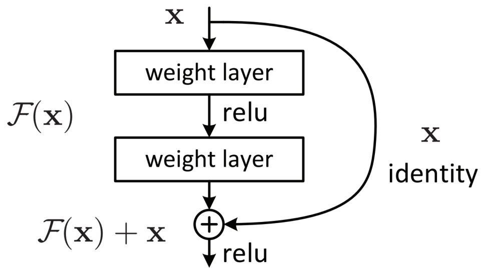


Figure B.5: Skip connection

B.2.2. UP-PROJECTION BLOCK

As discussed in Sec. III-B, we use modified up-projection blocks [16] in our roughness module. Up-projection blocks are the main constituent of the decoder in Fully Convolutional Residual Networks (FCRNs) which achieved state-of-the-art results in depth estimation from a single image. The key attribute of this block is the use of skip connection (see Fig. B.5), a simple and novel concept that put ResNet [6] at the forefront of image classification. Skip connection contains a short-cut branch that splits from and remerges with the main branch after skipping some layers. The short-cut branch can be with or without (identity) an additional convolutional layer.

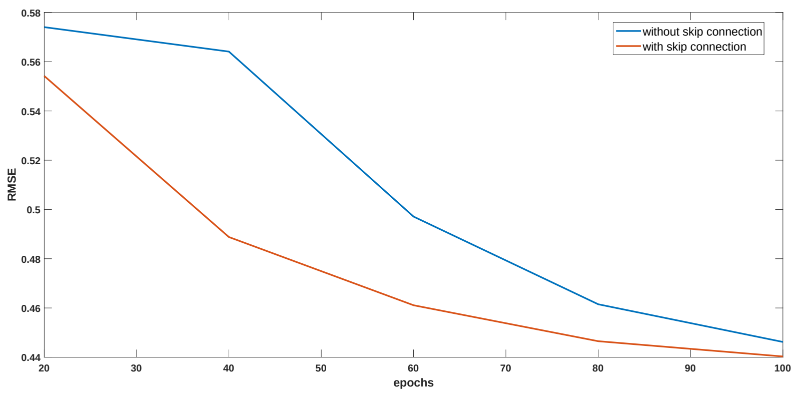


Figure B.6: Effect of skip connection. Faster reduction in the RMSE error can be observed with the skip connection

In a deep network trained with SGD optimizer, the backpropagated error signal is obtained through repeated multiplication of gradient updates. In case a layer approaches convergence, its gradient is small which, when multiplied, may cause the earlier layers to suffer from vanishing gradients. The short-cut branch in the skip connection offers an unimpeded path for backpropagation, thereby facilitating quick convergence [17]. Fig. B.6 shows a plot, which depicts the rate at which the error decreases, with and without the skip connection for our model.

Up-projection block positions the skip connection right after upsampling. The unpooling operation used in our method is different from that of FCRN. When an input element is expanded into a 2×2 patch, the element's location is always at the top left of the patch in FCRN while our method decides based on the pooling indices. The main branch of the skip connection in the block contains two convolutional layers, each followed by BN and ReLU, while the short-cut branch has a convolutional layer of its own. The first convolutional layer in both the branches has a receptive field of 5×5 which is changed to 3×3 initially for computational reasons. However, through experimentation, it is determined that a receptive field of 5×5 for the first convolutional layer in the main branch is essential as it captures the variation in roughness better than a receptive field of 3×3, as seen through b) and c) of Fig. B.7. We let the receptive field of the convolutional layer in the skip branch be 3×3 as a similar improvement is not observed.

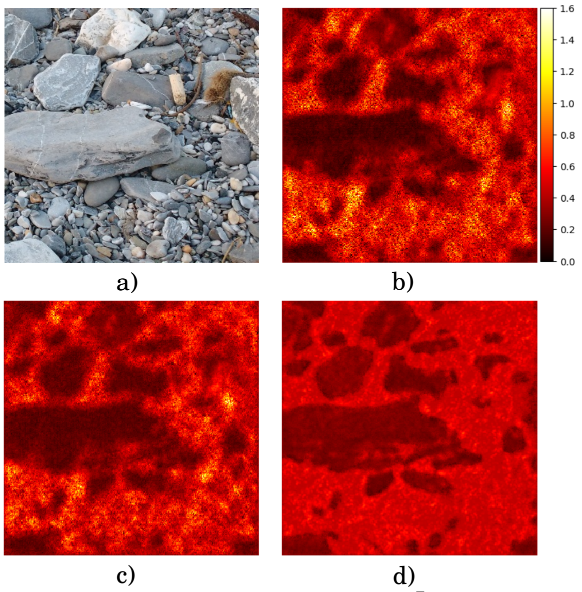


Figure B.7: a) input image, b) 5×5 kernel and learning rate of 10^{-7}, c) 3×3 kernel and learning rate of 10^{-7}, d) b) 5×5 kernel and learning rate of 10^{-6}

B.2.3. HYPER-PARAMETERS

The roughness module has to be trained from scratch for which a suitable learning rate is required. We experiment with different learning rates, starting from 10^{-3} at which the module fails to train because of gradient explosion caused by large errors. For every failed iteration, the learning rate is reduced by a factor of 10. At a learning rate of 10^{-6}, the module trains successfully for the first time. However, the estimated roughness is not so good. Albeit higher RMSE error than 10^{-6}, a learning rate of 10^{-7} produces superior prediction (compare b) and d) in Fig. B.7).

Apart from the learning rate, the module is trained with different optimizers (summarized in Table B.4) and SGD is found to give the best performance.

Optimizer	Loss	RMSE	RMSLE
Adam [18]	Berhu	0.5797	0.1559
	Euclidean	0.6650	0.1742
RMSProp[19]	Berhu	0.5579	0.1540
	Euclidean	0.5675	0.1618
SGD	Berhu	**0.4335**	**0.1119**
	Euclidean	0.4350	0.1123

Table B.4: Summary of validation loss for different optimizers

C
STRATEGIES USED IN OUR MODEL

Certain strategies, such as transfer learning, data augmentation, and median frequency balancing, are employed to improve the overall predictive power of the model. A discussion on these strategies is provided below,

C.1. TRANSFER LEARNING

Training the SegNet architecture from scratch is not feasible as the custom-built dataset is small in size. A common practice for CNNs is to initialize the model with pre-trained weights and finetune on the target dataset at a low learning rate. This principle, belonging to a broad subject called transfer learning or domain adaptation, uses the accumulated information gained from a source task with large datasets to learn a new task, in our case, terrain segmentation (See Fig. C.1). Greater the relationship between the source and target

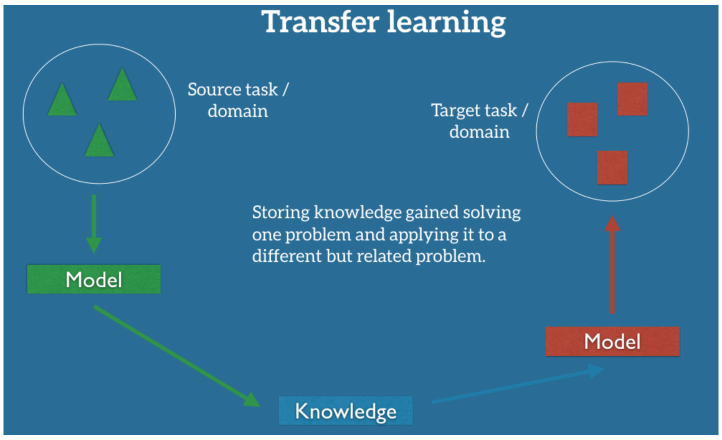

Transfer learning
Source task / domain
Target task / domain
Storing knowledge gained solving one problem and applying it to a different but related problem.
Model
Model
Knowledge

Figure C.1: Principle of transfer learning [20]

domain, better the transferability of knowledge. To assess this hypothesis, we fine-tune SegNet on our dataset with pre-trained weights from cityscapes [11] and SUN-RGBD [10] as initializers. It can be seen, from Fig. C.2, that cityscapes provides a better performance than SUN-RGBD as the latter is indoor-based dataset and less related to the terrain segmentation task.

One of the main challenges of transfer learning lies in deciding the number of convolutional layers that are to be fine-tuned. This depends on the similarity between the tasks and dataset size. For instance, if the dataset is quite small, only the last few layers are fine-tuned as the model's representation space increases in specificity with depth. The generic features will most likely remain the same for this small dataset. Moreover, the low number of samples is not a sufficient representation of the entire sample space and fine-tuning an entire network can lead to overfitting.

Given the size of our dataset, we fine-tune the entire SegNet initially. The trained model shows signs of overfitting on the test images. As a next step, the model is fine-tuned again after freezing the first layer

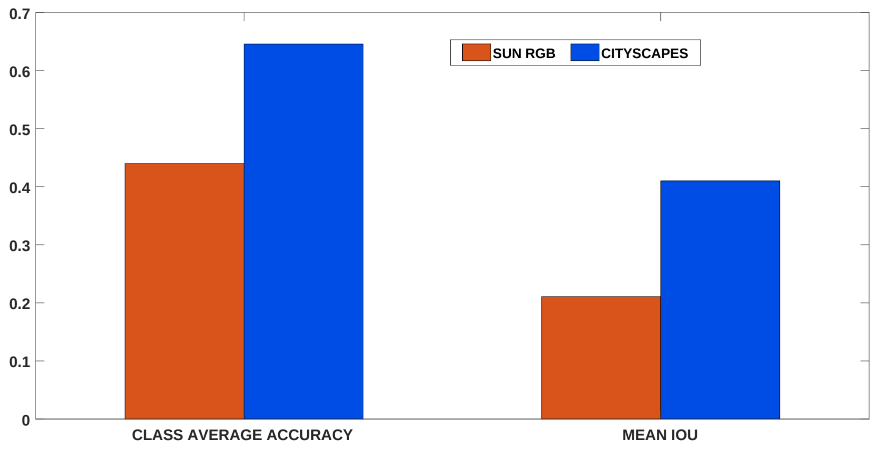


Figure C.2: Comparison between cityscapes and SUN-RGBD weights as initializer

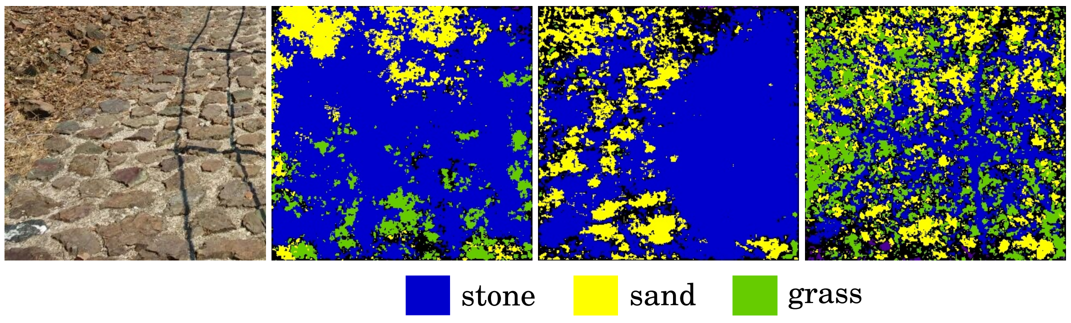


Figure C.3: Effect of fine-tuning. **From left to right:** a test image, fully fine-tuned, first layer in the model frozen, freezing the first two layers in the model

and an improvement in the prediction is observed. However, freezing the second convolutional layer results in an underfitted output. Fig. C.3 shows an example of a test image segmented with different fine-tuning configurations.

C.2. Data Augmentation

Given the limited training samples and variability in the outdoor environment, there can be instances where the robot observes a scene that is different from the samples in the training set. A way for the robot to handle these unseen situations is to improve the generalizability of the model. Data augmentation is one such technique that relies on basic image transformations to expose the model to different settings thereby making it more invariant to environmental changes. For example, rotation can alleviate biases related to the location of a particular terrain such as in cityscapes where the road is mostly present in the lower half of the image.

Two different transformations are used in our approach: horizontal/vertical flip and brightness/contrast adjustment. Both the implementations are on-the-fly and rely on inbuilt OpenCV functions. The flip function mirrors the input image along both the rows and columns. Small adjustments in the brightness and contrast are implemented through the following equation.

$$I = \alpha I_0 + \beta$$

where α is the contrast, β is the brightness, I_0 is the input image. The possible set of transformations is restricted to a range of values, $0.8, 0.9, ..., 1.2$ for contrast and $-6, -5, ..., 6$ for brightness. As mentioned in Sec. III-A., cropping is not used as the labelled region occupies a small portion in majority of the images. Hence, a random crop with most pixels unlabelled has an undesirable effect of vanishing gradients and increase in convergence time.

C.3. MEDIAN FREQUENCY BALANCING

Extracting a completely balanced dataset is difficult due to the complex mix of terrain classes in the training set images. As a result, a small imbalance exists in the dataset which is accommodated through median frequency balancing [21]. Through this balancing, the classes with frequency lesser than the median frequency have significantly higher weights than the most occurring classes. The weights for each class is computed using the following formulation,

$$weight(i) = \frac{median(freq_1, ...freq_k)}{freq_i}$$

where

$$freq_i = \frac{\sum_{j=1}^{I} p_j}{\sum_{i=1}^{k} \sum_{j=1}^{I} p_{ij}}$$

For a given image j, p_j refers to the number of pixels belonging to class i. $freq_i$ is the frequency of a particular class i, which is equal to pixels belonging to class i in the entire dataset divided by the total number of pixels.

D
Datasets and Evaluation metrics

This appendix provides details on the construction of datasets and the evaluation metrics used for each sub-task.

D.1. Segmentation datasets

A dataset of terrain images and corresponding labels is required for training CNNs. We consider various open-source segmentation datasets, two of which are used for building the terrain segmentation dataset. These are elaborated below,

1. **Cityscapes:** Cityscapes [11] is a large dataset that focusses on the urban street scenes from various cities. It provides pixel-wise annotations for 34 commonly found classes in street scenes, for example, road, building, tree etc. The finely annotated dataset consists of 2975 training and 500 validation images. The images are basically frames extracted from videos recorded by a stereo camera on the car while moving around the cities. For our objective, this is less interesting as the entire dataset has a constant environmental setting.

2. **PASCAL-Context:** Pascal-Context [22] provides the pixel-wise annotations to the images in PASCAL VOC 2010 challenge. The dataset includes a large number of classes from variety of scenes. However, only 59 classes are significantly frequent while the rest are sparse. The training and validation set contains 10103 images along with its annotations.

3. **ADE-20K:** ADE20K [23], a rich and ever-increasing dataset, consists of more diverse scenes and labels than Pascal-Context. The dataset provides accurate and clear-cut annotations for 20210 training and 2000 validation images. Given the variability and large number of images, we use this dataset for our purpose.

4. **Opensurfaces:** Opensurfaces [24] is a dataset with a focus on commonly found materials in indoor environment. It contains 25357 training images, taken from Flickr, and labels with annotations from 23 material categories. Despite being a huge dataset, it lacks predominant outdoor terrain classes such as asphalt and sand. However, wood and metal, which are sparsely found in ADE20K, is abundantly available. This necessitates us to combine Opensurfaces and ADE20k to construct the terrain segmentation dataset. The statistics of the training set is shown in Fig D.1.

The relevant labels from the respective dataset are replaced by a number in the range of $0-6$, with each number representing a terrain class as shown in Table D.1. 6 refers to the unlabelled or irrelevant classes which is ignored while calculating the training loss. A training iteration takes a mini-batch of images and labels as input from the randomly shuffled dataset. Shuffling, which is implemented in real-time, is important in eliminating biases or patterns in the dataset.

D.2. Roughness from Depth Image

We use the depth images taken from GeoMat [25] as the basis for creating ground-truth roughness values. The details of the GeoMat dataset is already provided in Sec. IV-A.2. The ground-truth roughness values, in

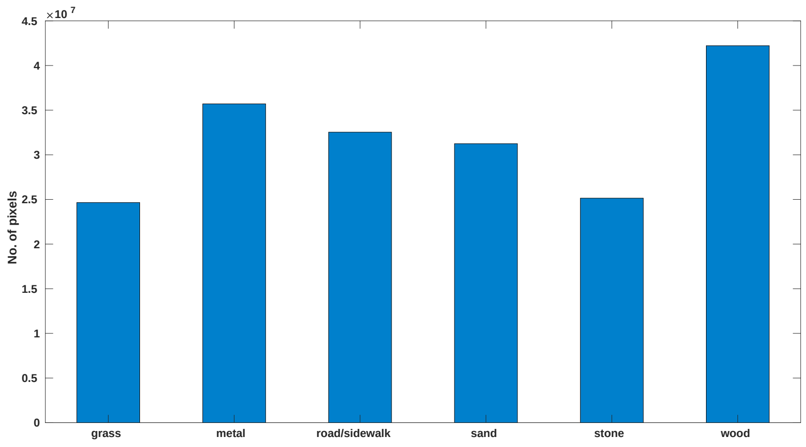


Figure D.1: Pixel count for each class in the training set

Sand	Wood	Stone	Metal	Road/Sidewalk	Grass	Irrelevant classes
0	1	2	3	4	5	6

Table D.1: Terrain classes and its label

our approach, are described through the point to plane distance, which depicts the deviations of the surface. Given a point cloud, the underlying idea is to fit a plane on a set of points in the neighbourhood of a specific point. The absolute distance between that particular point and the fitted plane gives the roughness value at that point.

As a first step, we require point clouds, which are generated from the depth images. The z_i (depth) coordinate at each pixel in the depth image is transformed to x_c,y_c,z_c coordinates of the point clouds using the intrinsic matrix given in the dataset. Basically, the transformation is from the image to the camera frame. The depth remains the same while x and y are given by,

$$x_c = \frac{x_i - x_0}{f_x} z_i$$

$$y_c = \frac{y_i - y_0}{f_y} z_i$$

$$z_c = z_i$$

where $\{f_x, f_y\}$ is the focal lengths and $\{x_0, y_0\}$ is the principal point of the camera. $\{x_i, y_i\}$ are the pixel coordinates of the depth image. An organized point cloud is obtained after transforming the entire depth image. As mentioned in Sec. IV-A.2, the plane is fitted on nine nearest neighbouring points. Increasing the number of neighbouring points exaggerates the roughness in the terrain patch.

A plane can be determined from a reference point and its normal. In the plane equation,

$$ax + by + cz + d = 0$$

$$d = -(ax_r + by_r + cz_r)$$

where $n = \{a, b, c\}$ is the normal and $P_r = \{x_r, y_r, z_r\}$ is the reference point. As the plane is fitted on the neighbouring patch, the reference point is taken as the mean of the patch, given by,

$$P_r = \frac{1}{m} \sum_{i=1}^{m} P_i$$

Here, m = 9. To compute the normal, PCL has an inbuilt function that minimizes the sum of squared distances (SSD) between each point and the plane.

$$\min \sum_{i=1}^{m} [(P_i - P_r)^T n]^2$$

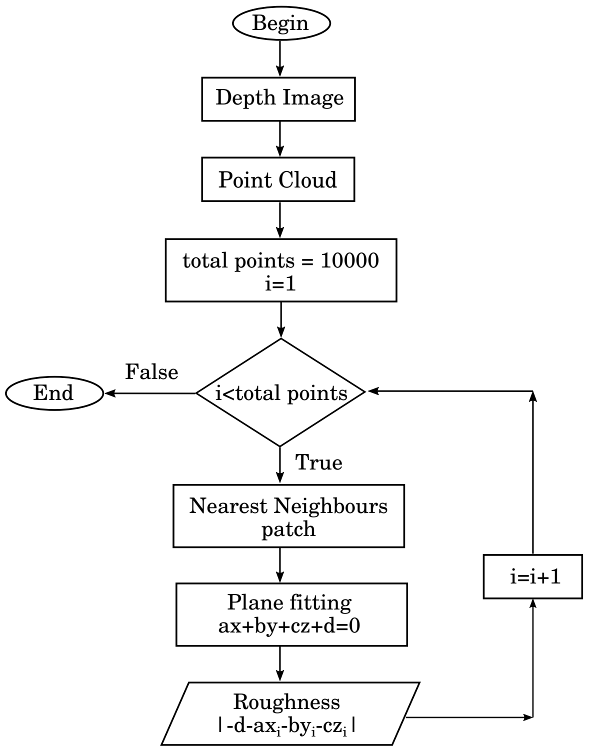


Figure D.2: Depth image to Roughness

This minimization problem reduces to minimizing the covariance matrix which is equivalent to finding the eigen vector corresponding to the smallest eigen value. Expanding the above equation,

$$\min \sum_{i=1}^{m} n^T (P_i - P_r)(P_i - P_r)^T n$$

where the covariance matrix,

$$C = \frac{1}{m} \sum_{i=1}^{m} (P_i - P_r)(P_i - P_r)^T$$

Solving for the eigen vectors, we determine the normals n. The plane parameters are calculated with the reference point and normal. As we require per-pixel roughness, the absolute distance from each point to the plane is computed. The algorithm is summarized in Fig. D.2.

D.3. EVALUATION METRICS

D.3.1. PERFORMANCE MEASURE FOR SEGMENTATION

To evaluate the semantic segmentation performance, we employ two commonly used metrics for evaluating semantic segmentation: class average accuracy and mean IoU. Class average accuracy, a measure that accounts for the correctly classified pixels in each class, is given by,

$$\frac{1}{k} \sum_{i=1}^{k} \frac{TP_i}{TP_i + FN_i}$$

where k is the total number of classes, TP_i is the true positives and FN_i is the false negatives for a given class. Mean IoU is the most used metric for semantic segmentation as it is simple, yet a more precise measure than accuracy. Apart from true positives (TP) and false negatives (FP), this measure includes false positives (FP), thereby providing a finer description of the performance. It is given by,

$$\frac{1}{k} \sum_{i=1}^{k} \frac{TP_i}{TP_i + FN_i + FP_i}$$

D.3.2. ERROR METRICS FOR ROUGHNESS

The regressed roughness is evaluated on the validation set using Root Mean Squared Error (RMSE) and Root Mean Squared Log Error (RMSLE). RMSE is the one of the widely used metrics for regression methods. The square in the expression, shown below, prevents the cancellation of positive and negative errors and gives higher weightage to large errors.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(predict_i - actual_i)^2}{N}}$$

where N is the total number of pixels in the image.

RMSLE is slightly different from RMSE in the sense that it computes the ratio of predicted to the ground-truth due to the presence of log. It weighs the under-estimated predictions more than over-estimated predictions. The formula for RMSLE is given below,

$$RMSLE = \sqrt{\frac{\sum_{i=1}^{N}(\log(1 + predict_i) - \log(1 + actual_i))^2}{N}}$$

Bibliography

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition,* Proceedings of the IEEE **86**, 2278 (1998).

[2] *CS231n: Convolutional Neural Networks for Visual Recognition,* `http://cs231n.github.io/convolutional-networks/#layers`.

[3] X. Glorot, A. Bordes, and Y. Bengio, *Deep Sparse Rectifier Neural Networks,* in *Proc. of the Fourteenth International Conference on Artificial Intelligence and Statistics*, Vol. 15 (2011) pp. 315–323.

[4] S. Ioffe and C. Szegedy, *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,* in *Proc. of the 32nd International Conf. on International Conference on Machine Learning - Volume 37* (2015) pp. 448–456.

[5] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition,* CoRR **abs/1409.1556** (2014).

[6] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition,* arXiv preprint arXiv:1512.03385 (2015).

[7] J. Long, E. Shelhamer, and T. Darrell, *Fully Convolutional Networks for Semantic Segmentation,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015) pp. 3431–3440.

[8] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, *ENet: A Deep Neural Network Architecture for Real-Time Semantic Segmentation,* CoRR **abs/1606.02147** (2016), arXiv:1606.02147 .

[9] G. J. Brostow, J. Fauqueur, and R. Cipolla, *Semantic Object Classes in Video: A High-Definition Ground Truth Database,* Pattern Recognition Letters (2008).

[10] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, *Learning Deep Features for Scene Recognition using Places Database,* in *Advances in Neural Information Processing Systems 27* (2014) pp. 487–495.

[11] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, *The Cityscapes Dataset for Semantic Urban Scene Understanding,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[12] A. Garcia-Garcia, S. Orts, S. Oprea, V. Villena-Martinez, and J. G. Rodríguez, *A Review on Deep Learning Techniques Applied to Semantic Segmentation,* CoRR **abs/1704.06857** (2017).

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *ImageNet Classification with Deep Convolutional Neural Networks,* in *25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12 (Curran Associates Inc., USA, 2012) pp. 1097–1105.

[14] S. Lathuilière, P. Mesejo, X. Alameda-Pineda, and R. Horaud, *A Comprehensive Analysis of Deep Regression,* CoRR **abs/1803.08450** (2018).

[15] F. Ma and S. Karaman, *Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image, ICRA,* (2018).

[16] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab, *Deeper Depth Prediction with Fully Convolutional Residual Networks,* in *Fourth International Conf. on 3D Vision (3DV)* (2016) pp. 239–248.

[17] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadoury, and C. J. Pal, *The Importance of Skip Connections in Biomedical Image Segmentation,* in *LABELS/DLMIA@MICCAI* (2016).

[18] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization,* CoRR **abs/1412.6980** (2014).

[19] G. Hinton, N. Srivastava, and K. Swersky, *Lecture 6a Overview of Mini–Batch Gradient Descent,* Coursera Lecture slides https://class.coursera. org/neuralnets-2012-001/lecture (2012).

[20] *Transfer Learning - Machine Learning's Next Frontier,* http://ruder.io/transfer-learning/.

[21] D. Eigen and R. Fergus, *Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture,* in *IEEE International Conference on Computer Vision (ICCV)* (IEEE Computer Society, Washington, DC, USA, 2015) pp. 2650–2658.

[22] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, *The role of context for object detection and semantic segmentation in the wild,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[23] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, *Semantic Understanding of Scenes Through the ADE20K Dataset,* arXiv preprint arXiv:1608.05442 (2016).

[24] S. Bell, P. Upchurch, N. Snavely, and K. Bala, *OpenSurfaces: A Richly Annotated Catalog of Surface Appearance,* ACM Trans. on Graphics (SIGGRAPH) **32** (2013).

[25] J. DeGol, M. Golparvar-Fard, and D. Hoiem, *Geometry-Informed Material Recognition,* in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).