CNN Robotic Vision for Ground Segmentation and Motion Decision

U-Net and DeepLabv3 applicability for ground segmentation and motion decisions

Virginia Ceccatelli (261205098) - U-Net
Haorui Li (261085981) - DeepLabv3
August 2025

Table of Content

Abstract

As mobile robotics continues to evolve, lightweight and low-cost vision systems are becoming increasingly critical for simulating and enhancing real-time navigation. This project presents a vision-based ground segmentation and motion planning framework tailored to indoor robotic applications, specifically in hospital settings. Using a U-Net architecture with a MobileNetV2 backbone, the code classifies traversable versus non-traversable ground from a computer-camera input. It then computes optimal steering angles based on live camera feeds. The project supports inference on images, videos, and live camera input, with the latter available in both Python and C++ to balance adaptability (Python) and efficiency (C++). The system outputs real-time decision-making through segmentation masks and probabilistic heatmaps that visualize model confidence, thereby aiding the robot's understanding of its surroundings. This work lays the foundation for a vision-based, cost-efficient autonomous navigation module for robots in indoor hospital environments, with potential for future integration with depth cameras, external processors and global localization systems.

Keywords: Robotic Vision, Ground Segmentation, Motion Planning, Autonomous Navigation, U-Net, MobileNetV2, Real-Time Inference, Hospital Robotics.

Introduction

Rising demands for adaptable and budget-friendly industrial robotics have led to advancements in visual perception tools that enable robots to analyze and process their surroundings in real time. Traditional solutions such as LiDAR or depth cameras, though powerful, can be expensive and complex. A robotic vision and motion decision simulation tool is essential for safely and efficiently bridging perception and control in a robot operating system, enabling interpretable, and reliable navigation decisions before real-world deployment.

This project works as an adaptive perception system that remains computationally lightweight, by developing a deep learning-based ground segmentation system running solely on a computer OS - without need of extra processors (while still allowing easy future integration). The system distinguishes traversable surfaces from obstacles using a U-Net model with a MobileNetV2 encoder.

The project workflow includes data labeling via LabelMe, augmentation-enhanced training, and evaluation of performance using various metrics such as intersection-over-union (IoU). To enhance usability, inference modules were developed for static images, video streams, and live camera feeds. The model analyzes the segmentation mask in real-time to determine the safest navigation path, calculated by identifying the region with the highest density of predicted ground pixels. The result is a responsive, interpretable, and deployable module that outputs both clear segmentation masks and confidence-based heatmaps. The live-feed codebase has been translated into C++ for faster/ better performance while preserving the Python version for flexibility in prototyping and future adjustments.

Lastly, the potential of DeepLabv3 was explored as a comparative model to U-Net; however, it demonstrated significantly lower accuracy in recognizing the ground plane. Future work could focus on improving its effectiveness for ground segmentation and robotic motion decision tasks.

Theory

This project is grounded in the theory of semantic segmentation, a computer vision task in which each pixel in an image is assigned to a predefined category. In this case, the model classifies pixels as either “ground” or “not ground” to support motion planning for robotic navigation. The core model architecture is a U-Net with a MobileNetV2 encoder, pre-trained on the ImageNet dataset. U-Net is a convolutional neural network designed for dense, pixel-level prediction, featuring a symmetric encoder-decoder structure: the encoder compresses spatial information to extract high-level features, while the decoder reconstructs the spatial resolution through upsampling. MobileNetV2, used as the encoder, is a lightweight and computationally efficient architecture ideal for real-time inference. Because it is pre-trained on ImageNet, it starts with general visual feature representations, leading to faster convergence and better performance when fine-tuned on small, domain-specific datasets like the one used here.

Literature Review

Recent advances in robotic vision have enabled significant progress in autonomous navigation, particularly in structured indoor environments like hospitals. However, most existing systems still rely on expensive hardware such as LIDAR and high-performance GPUs, making them less accessible for lightweight or cost-sensitive applications. For example, Kim et al. (2022)[1] developed DPoom, combining an RGB-D camera with plane segmentation and SLAM, but it offered little cost advantage. In contrast, Jo et al. (2022)[2] proposed SMARTmBOT, a cheaper alternative using a Raspberry Pi and ToF sensors, but lacking visual understanding. Other projects, such as those using YOLO (You Only Look Once) with reinforcement learning or event-based cameras (CVPRW 2025[3]), demonstrate advanced obstacle avoidance but require large datasets and training infrastructure, limiting accessibility.

This project distinguishes itself by using a MobileNetV2-based U-Net for pixel-wise ground segmentation using only RGB input, achieving semantic understanding on low-cost hardware without reliance on depth sensors or GPUs.

Dataset Preprocessing

A total of 100 images were manually annotated using a tool called LabelMe, a graphical annotation program used to define polygonal regions - in this case, "ground" and "not_ground" - and save them as .json files. A script (convert_masks.py) then converts these .json files into binary masks by assigning a value of 1 to “ground” areas and 0 to “not_ground” areas. It also ensures that any overlapping “not_ground” regions overwrite “ground” labels, so that obstacles are not mistakenly learned as traversable ground during training.

The model used is a U-Net with a MobileNetV2 encoder, a lightweight and efficient CNN architecture commonly used for

[1] Kim, Taekyung & Lim, Seunghyun & Shin, Gwanjun & Sim, Geonhee & Yun, Dongwon. (2022). An Open-Source Low-Cost Mobile Robot System with an RGB-D Camera and Efficient Real-Time Navigation Algorithm. IEEE Access. PP. 1-1. 10.1109/ACCESS.2022.3226784.

[2] Jo, W., Kim, J., Wang, R., Pan, J., Senthilkumaran, R. K., & Min, B. C. (2022). Smartmbot: A ros2-based low-cost and open-source mobile robot platform. arXiv preprint arXiv:2203.08903.

[3] Bugueno-Cordova, I., Ruiz-del-Solar, J., & Verschae, R. (2025). Human-Robot Navigation using Event-based Cameras and Reinforcement Learning. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 5004-5012).

feature extraction in classification and segmentation tasks. The Adam optimizer (Adaptive Moment Estimation) was chosen for its efficiency and ease of tuning, while PyTorch's built-in BCEWithLogitsLoss was used as the loss function due to its strong performance for binary classification tasks; binary cross-entropy loss penalizes confident but incorrect predictions more heavily, encouraging the model to become more accurate over time.

Algorithm Development

The algorithm development process began with training a U-Net model configured with a MobileNetV2 encoder pretrained on ImageNet. This encoder acts as a lightweight yet efficient feature extractor, while the U-Net decoder reconstructs dense segmentation maps from the compressed features. From the start, the model was trained with single-channel binary masks indicating ground versus non-ground regions. Early experiments focused on iteratively refining the model through small changes each training cycle, especially in the data preprocessing and augmentation stages.

To improve the model's generalization ability, image augmentations were added using the 'Albumentations' library. These augmentations artificially increased dataset diversity, reducing overfitting and improving robustness to lighting and perspective changes. Specifically, the augmentation pipeline included HorizontalFlip applied with 60% probability, RandomBrightnessContrast (50%), and ToGray (10%). These choices were informed by visual inspection of the dataset, which featured diverse lighting conditions, reflective surfaces, and asymmetric environments.

For normalization, each image was scaled using the ImageNet mean and standard deviation, ensuring zero-centered inputs for faster convergence and more stable gradients. This was particularly important given the encoder's pretraining on ImageNet, aligning input statistics with its original training regime.

Various training hyperparameters were tuned across multiple runs. After testing several combinations, a batch size of 8 and training over 40 epochs with a learning rate of either 0.001 or 0.0001 using the Adam optimizer proved most effective. The model was evaluated using validation loss and Intersection over Union (IoU), providing insight into pixel-wise accuracy and overlap between predicted and ground truth masks.

During training, it became evident that the model initially struggled with paths containing small, standalone obstacles located in the center of the ground path—a pattern underrepresented in the early dataset. To address this, new images were labeled and added, and a logic layer was introduced during preprocessing to subtract 'not_ground' masks from 'ground' masks to avoid labeling obstacles as traversable. This ensured that the model learned to recognize not only ground surfaces as a general area, but specifically which ground surface is traversable and which is not traversable/ safe.

Visual inspection also revealed that the model occasionally made overconfident predictions in ambiguous areas such as shiny surfaces, shadows, or occluded regions. These challenges led to further tuning of the augmentation strategy and validation of model checkpoints.

Ultimately, the best-performing and safest model achieved a validation loss of 0.1177 and a validation IoU of 0.8629, marking a substantial improvement from early training cycles. Although previous models achieved better validation metrics, they were either overconfident in shady and shiny areas, or they generally did not behave safely.

Experimental Method

The experiments were conducted on a laptop with a webcam, and later tested on video feeds and live camera streams. All experiments were executed through a command-line interface.

To train a new model:

1. Install all dependencies and requirements (requirements.txt)
2. Save and Label ~100 images using LabelMe
3. Follow the steps on GitHub to train the model

To use and test the developed model:

1. Install all dependencies and requirements (requirements.txt)
2. Download the exported model
3. Follow the steps on GitHub to use the model inference scripts

Image and Video Input

At first, the model was simply tested on the validation dataset images and its predictions were visualized to understand shortcomings and possible improvements. Then, a short script for video inference (seg_video) was developed, where a video can be passed in as input instead of a simple image, and the model will overlay a moving/ adjusting ground plane mask prediction. This script already included a directional line indicating the "safest motion decision", simply calculated as a central line through the binary ground segmentation mask.

Live Feed Input

The latest script is able to take a live feed as input (tested on laptop camera, but it can be altered to any other live feed input, e.g. depth camera) and overlay a predicted ground plane mask, with a more precise steering angle in positive or negative degrees, visualized as a line. This is processed by dividing the bottom half of the screen into small segments (num_beams), where the number of "ground" pixels in each are counted, and the segment with the highest amount of "ground" is determined to be the momentary best motion decision (updated every 3 seconds). The live feed script also outputs another video showing a "heatmap" instead of a clear ground plane mask, where the colour palette changes based on the classification probability, i.e. how confidently the model assigned that area as "ground" vs "not_ground". Red is assigned to the highest probabilities and can thus be interpreted as the safest area, whereas yellow, green and blue are stage-wise less safe. This script was deemed to be the most useful and applicable to the robot project, so the initial Python code was translated to C++, keeping both; The Python code is much easier to adapt and alter, as all the modules used (especially OpenCV) are more permissive in Python, but the C++ code seems to run slightly faster, and can thus be helpful for robotics simulations.

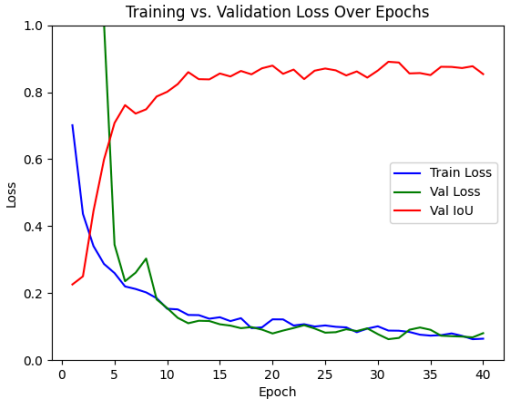
Experimental Results

This is a subset of validation performance of the best epoch of different training cycles:

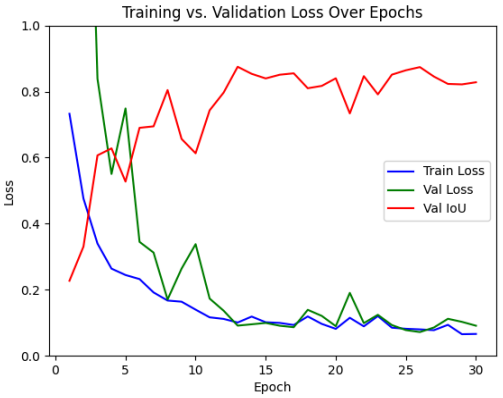
Num	**Val Loss**	**Val IoU**	**Comment**
1	0.0627	0.8907	Sharp predictions, some overconfident errors, did not recognize small stand-alone obstacles
2	0.0608	0.8926	Best accuracy and caution but did not recognize small stand-alone obstacles
3	0.0719	0.8740	Misclassified shadows as ground
4	0.0594	0.8940	Too confident; less

			safe in hospitals
5	0.0530	0.9056	Best IoU, but less conservative in behavior, did not recognize small stand-alone obstacles
6	0.1177	0.8629	Recognized in-path obstacles and was relatively cautious

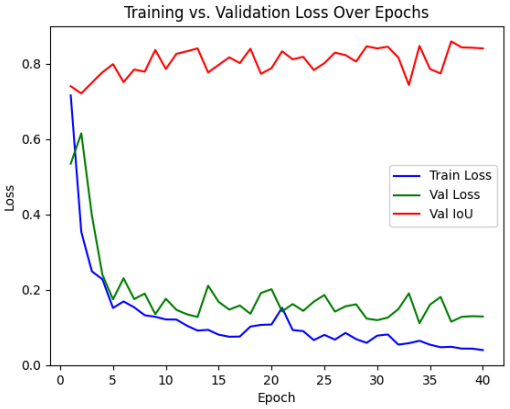
Taking a closer look at model training; model one showed very rapid initial improvement, with validation IoU rising sharply by epoch 5 and peaking early. However, after the initial gains, the validation IoU plateaued and even began to slightly decline, while training loss continued decreasing. This suggests the model may have started to overfit early, especially due to limited data variation or overconfident predictions in ambiguous regions.



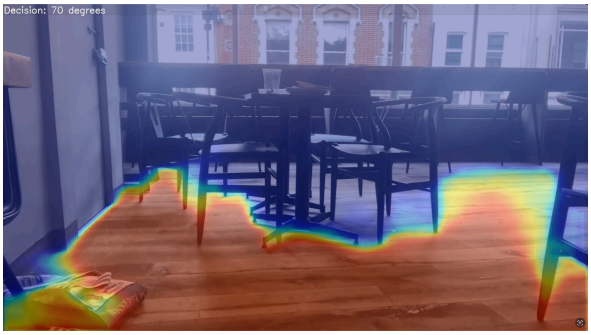
Model 3 showed strong overall learning, with validation IoU reaching high values (>0.85), but also significant fluctuations across epochs. Although it performed well in terms of peak metrics, its volatility posed a risk for deployment, in terms of stability and safety.



The final model was selected due to its consistently improving validation loss and high IoU, which remained stable over time. While it did not achieve the absolute highest IoU compared to some earlier models, it provided the best balance between precision and safety.



Checkpoint 6 was thus selected for deployment based on safety - it recognized obstacles relatively better than the other checkpoints and it was not overly confident. The classification avoided edge regions and ‘blue’ zones (in terms of the heatmap), it did not wrongfully classify ambiguous shadows or reflections, and it stayed relatively centered in unobstructed ground. Although the previous checkpoints might seem more successful, visual inspection revealed that they did not recognize small objects in the middle of traversable ground, posing a serious risk.



DeepLabv3

DeepLabv3 is a state-of-the-art semantic segmentation architecture that leverages atrous convolution and atrous spatial pyramid pooling (ASPP) to capture multi-scale contextual information without downsampling spatial resolution excessively (Chen et al., 2017). In theory, this architecture is well-suited for indoor floor segmentation tasks due to its ability to aggregate features over varying receptive fields and preserve localization accuracy. However, experimental results in this project revealed several limitations.

DeepLabv3 was fine-tuned using RGB images and binary masks extracted from the ADE20K dataset. The training process involved multiple trials, including up to 150 epochs, normalized input preprocessing, and resized target masks. However, some of the output visualizations during inference consistently yielded black masks—indicating the model failed to classify floor regions as expected. The failure likely occurred from several compounding issues. First, inference and training were conducted entirely on CPU, severely limiting the batch size, training speed, and convergence quality. DeepLabv3, while powerful, is computationally intensive and optimized for GPU acceleration. Second, label imbalances in the dataset and inconsistencies in floor label indices likely caused the model to generalize poorly. Another factor was the incorrect initial processing of segmentation masks. The original masks from ADE20K contain RGB-encoded labels, but were initially treated as single-channel class indices. Due to the lack of appropriate conversion or color-to-label mapping, the model was essentially performing misaligned supervised learning. This led the loss function to behave incorrectly and inhibited the model's ability to learn meaningful class boundaries during training.

Conclusion

This project successfully developed a ground segmentation and motion decision system that can be run entirely on simple laptop OS, without need for any other materials. The system achieved reliable performance across static, video, and live inputs and was successfully deployed in both Python and C++ runtimes. While multiple model configurations were tested, the selection criteria prioritized accuracy as well as safety - essential in busy/ highly cluttered environments such as hospital scenarios.
On the other hand, despite DeepLabv3's strong theoretical advantages, its performance in this project was limited due to computational constraints, incorrect mask preprocessing, and domain-specific dataset challenges.

Further improvements for the U-Net model include integration with monocular depth estimation to distinguish flat shadows from actual depressions, integrating it with a processor for faster decisions, or SLAM integration. If the project is to be kept as a lightweight and cost-efficient simulation tool, more images can be collected, labelled and the model can be trained and fine tuned to output better segmentation predictions than the

current best model. All the code and the exported model is on GitHub.

Further improvements for the DeepLabv3 model could deploy training on a GPU-supported infrastructure to allow for larger batch sizes and faster convergence.

References

Kim, Taekyung & Lim, Seunghyun & Shin, Gwanjun & Sim, Geonhee & Yun, Dongwon. (2022). An Open-Source Low-Cost Mobile Robot System with an RGB-D Camera and Efficient Real-Time Navigation Algorithm. IEEE Access. PP. 1-1. 10.1109/ACCESS.2022.3226784.

Jo, W., Kim, J., Wang, R., Pan, J., Senthilkumaran, R. K., & Min, B. C. (2022). Smartmbot: A ros2-based low-cost and open-source mobile robot platform. arXiv preprint arXiv:2203.08903.

Bugueno-Cordova, I., Ruiz-del-Solar, J., & Verschae, R. (2025). Human-Robot Navigation using Event-based Cameras and Reinforcement Learning. In Proceedings of the Computer Vision and Pattern Recognition Conference (pp. 5004-5012).

Ravasconcelos. (n.d.). ravasconcelos/rl_obstacle_avoidance: Robot obstacle avoidance with reinforcement learning. Retrieved from GitHub - ravasconcelos/rl_obstacle_avoidance: Robot obstacle avoidance with reinforcement learning

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI) (pp. 234–241). Springer.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18 (pp. 234-241). Springer international publishing.

K, B. (2024). U-Net Architecture For Image Segmentation. https://www.digitalocean.com/community/tutorials/unet-architecture-image-segmentation