Floor Detection using Support Vector Machines

A Comparative Study of Three Feature Engineering Approaches

C Murali Madhav
Enrollment Number: 230115

Advanced Machine Learning
Newton School of Technology
Rishihood University

February 2026

Abstract

This report presents a comprehensive study on floor detection in indoor images using Support Vector Machines (SVM). Three distinct feature engineering approaches were implemented and compared: RGB-only pixel classification, RGB with spatial coordinates, and KMeans-based region-level classification. Using the CMM dataset (my own custom dataset) with COCO-format annotations, I trained RBF kernel SVMs on 12 indoor images with an 80/20 stratified train-test split. Results demonstrate that the region-level KMeans approach achieves the highest test accuracy of approximately 92.5% with significantly reduced training time (0.008 seconds), outperforming both pixel-based methods. The RGB plus spatial features method achieved 86.1% accuracy, while the baseline RGB-only approach reached 78.0% accuracy. This study highlights the importance of feature engineering and the effectiveness of region-based approaches for semantic segmentation tasks.

1. Introduction

1.1 Background

Floor detection is a fundamental task in computer vision with applications in robotics, autonomous navigation, and indoor scene understanding. Accurately identifying floor regions in images enables robots to plan safe paths, helps visually impaired individuals navigate spaces, and supports augmented reality applications. Traditional approaches to floor detection have ranged from simple color-based segmentation to sophisticated deep learning models.

1.2 Problem Statement

The objective of this project is to classify image regions as floor or non-floor using ground-truth polygon annotations in COCO format. Rather than employing complex deep learning architectures, this study focuses on feature engineering combined with classical machine learning (SVM) to understand which features contribute most effectively to floor detection.

1.3 Objectives

The primary objectives of this study are:

- Implement three distinct feature extraction methods for floor detection
- Train and evaluate RBF kernel SVM classifiers for each method
- Compare accuracy, training time, and computational efficiency
- Analyze the trade-offs between pixel-level and region-level approaches

2. Dataset and Preprocessing

2.1 CMM Dataset

The CMM dataset consists of custom indoor images with floor annotations in COCO-style JSON format. The dataset includes polygon segmentations that precisely delineate floor regions in each image. Only images containing at least one floor annotation were utilized, resulting in 12 image-mask pairs for training and evaluation.

2.2 Preprocessing Steps

- **Image Resizing:** All images were resized so that the longer dimension does not exceed 512 pixels, maintaining aspect ratio to ensure manageable memory usage and computational speed
- **Mask Generation:** Binary masks were constructed by drawing polygon segmentations onto blank canvases, where floor pixels are labeled as 1 and non-floor pixels as 0
- **Region Selection:** For Methods 1 and 2, only the bottom half of each image was used, as floor regions typically appear in the lower portion of indoor photographs

3. Methodology

Three distinct feature engineering approaches were implemented to compare their effectiveness in floor detection. All methods employed an RBF kernel SVM with hyperparameters C=10, gamma='scale', and class_weight='balanced' to handle class imbalance. Features were standardized using StandardScaler before training.

3.1 Method 1: RGB-Only Pixel Classification

3.1.1 Approach

This baseline method uses only color information to classify pixels. Each pixel in the bottom half of the image is represented by its RGB values [R, G, B], forming a 3-dimensional feature vector. The classifier learns which color combinations typically correspond to floor surfaces (e.g., grey, brown, beige tones) versus non-floor regions.

3.1.2 Feature Extraction

- Extract bottom half of each image (rows from H/2 to H)
- For each pixel: features = [R, G, B]
- Subsample to maximum 5,000 pixels per image to limit dataset size
- Label = ground-truth mask value (0 = non-floor, 1 = floor)

3.1.3 Limitations

This approach cannot utilize spatial information. The model has no knowledge that floor regions typically appear at the bottom of images, limiting its ability to disambiguate similar colors in different image regions.

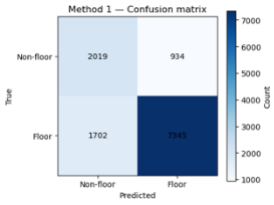
3.1.4 Related Images

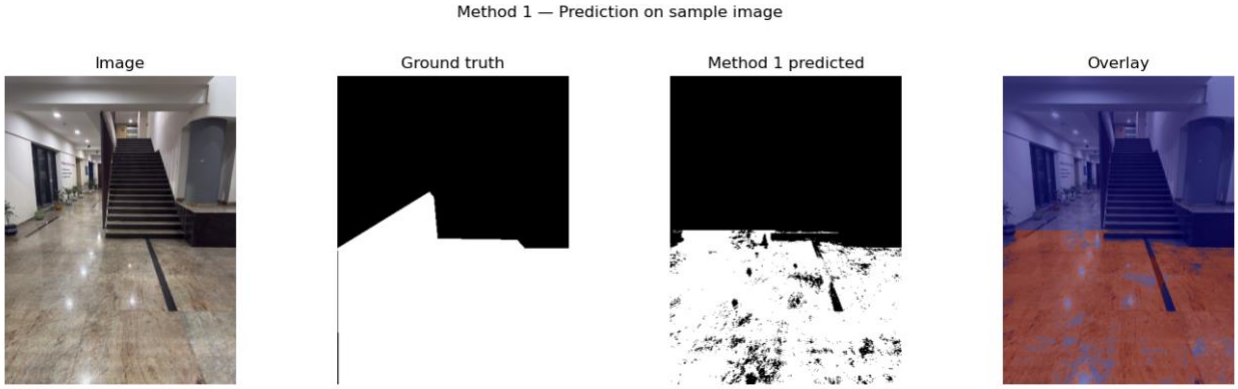

```
Method 1 — Classification report:
              precision    recall  f1-score   support

   Non-floor       0.54      0.68      0.61      2953
       Floor       0.89      0.81      0.85      9047

    accuracy                           0.78     12000
   macro avg       0.71      0.75      0.73     12000
weighted avg       0.80      0.78      0.79     12000

Method 1 — Accuracy: 0.7803333333333333
```





3.2 Method 2: RGB with Spatial Coordinates

3.2.1 Approach

Building upon Method 1, this approach incorporates pixel position within the image. By adding normalized x and y coordinates, the model can learn spatial patterns such as "floor regions tend to appear in the lower portion of the frame."

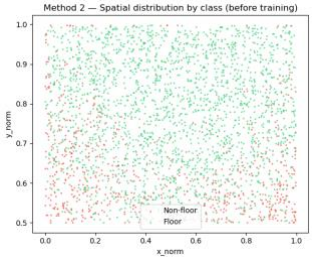
3.2.2 Feature Extraction

- Same region selection as Method 1 (bottom half of image)
- For each pixel: features = [R, G, B, x_norm, y_norm]
- x_norm = column_index / image_width (range: 0 to 1)
- y_norm = row_index / image_height (range: 0 to 1)

3.2.3 Advantages

The inclusion of spatial features allows the SVM to learn rules combining color and position, such as "darker pixels with high y_norm (bottom of image) are likely floor." This typically improves accuracy over purely color-based classification.

3.2.4 Related Images

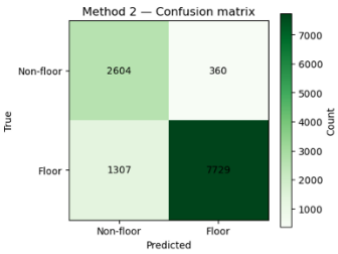


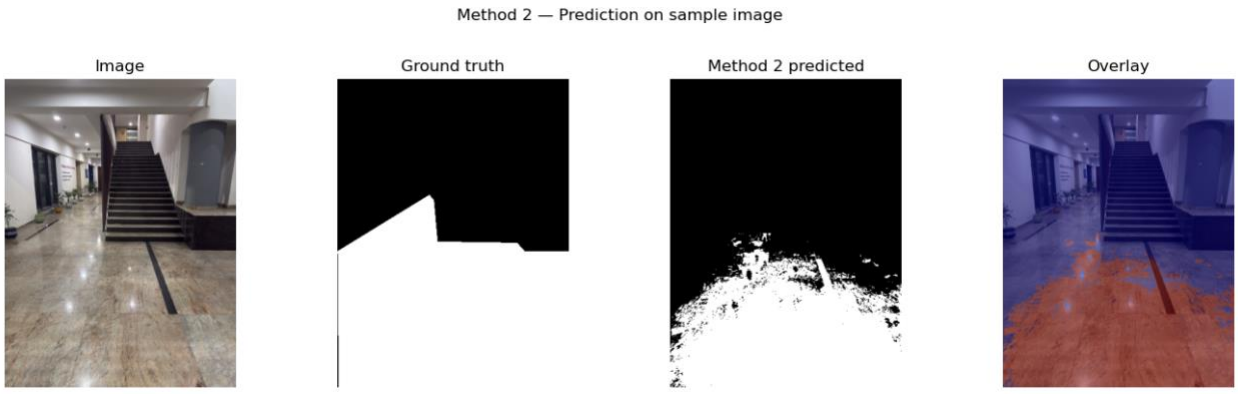

```
Method 2 — Classification report:
              precision    recall  f1-score   support

   Non-floor       0.67      0.88      0.76      2964
       Floor       0.96      0.86      0.90      9036

    accuracy                           0.86     12000
   macro avg       0.81      0.87      0.83     12000
weighted avg       0.88      0.86      0.87     12000

Method 2 — Accuracy: 0.8610833333333333
```





3.3 Method 3: KMeans Region-Level Classification

3.3.1 Approach

Rather than classifying individual pixels, this method groups similar pixels into coherent regions using KMeans clustering, then classifies entire regions. This reduces noise, decreases the number of training samples, and often improves both accuracy and computational efficiency.

3.3.2 Feature Extraction Process

Step 1 - KMeans Clustering:

- Represent all pixels in the full image as (R, G, B, x_norm, y_norm)
- Apply KMeans with n_clusters = 100 to assign each pixel a region ID

Step 2 - Region-Level Features:

- For each cluster/region, compute mean RGB values
- Calculate centroid position (cx_norm, cy_norm) in normalized coordinates
- Feature vector per region = [mean_R, mean_G, mean_B, cx_norm, cy_norm]

Step 3 - Region Labeling:

- Determine region label by majority vote of ground-truth mask pixels
- If > 50% of region's pixels are floor, label region as floor (1), else non-floor (0)

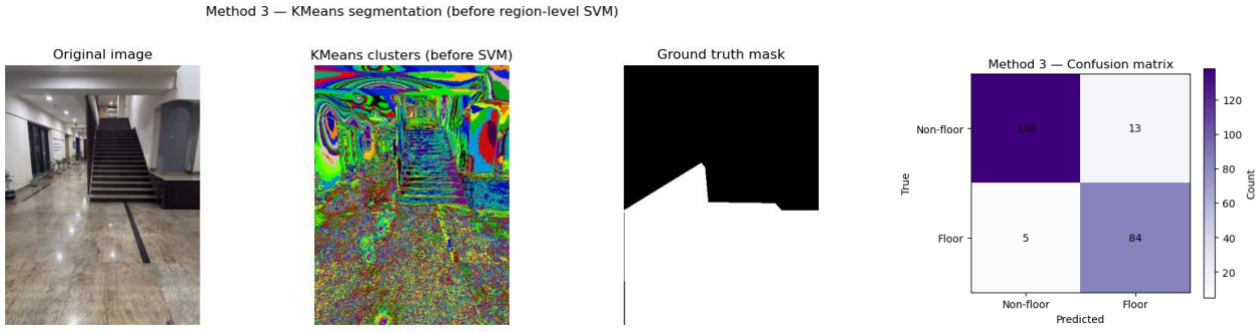
3.3.3 Dataset Characteristics

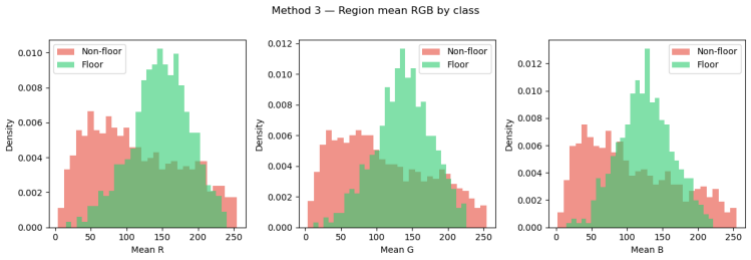
With 12 images and 100 regions per image, this method produces approximately 1,200 training samples (compared to tens of thousands of pixels in Methods 1 and 2). This dramatic reduction in sample size leads to significantly faster training while maintaining or improving accuracy.

3.3.4 Prediction and Visualization

After training, predictions are made at the region level and then mapped back to pixel-level masks for visualization. Each pixel inherits the classification of its assigned cluster, creating coherent segmented regions.

3.3.4 Related Images



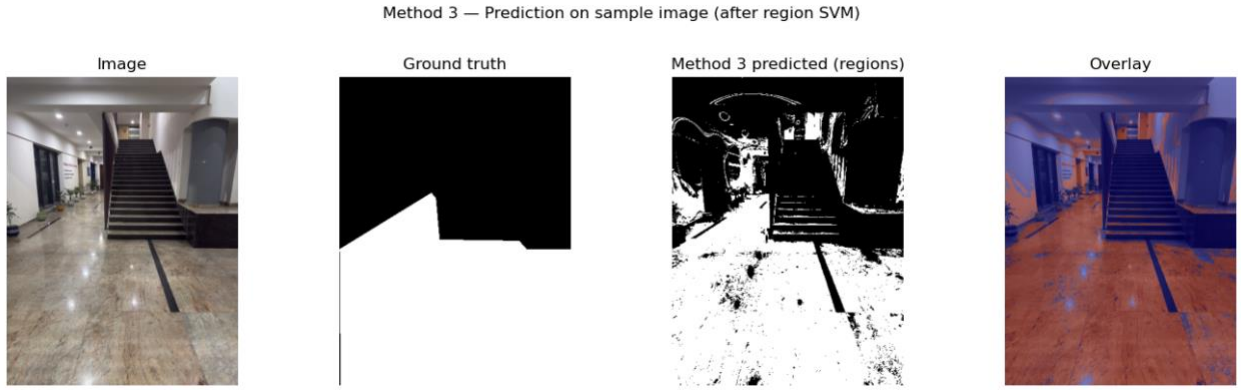



```
Method 3 — Classification report:
              precision    recall  f1-score   support

   Non-floor       0.97      0.91      0.94       151
       Floor       0.87      0.94      0.90        89

    accuracy                           0.93       240
   macro avg       0.92      0.93      0.92       240
weighted avg       0.93      0.93      0.93       240

Method 3 — Accuracy: 0.925
```



4. Results and Analysis

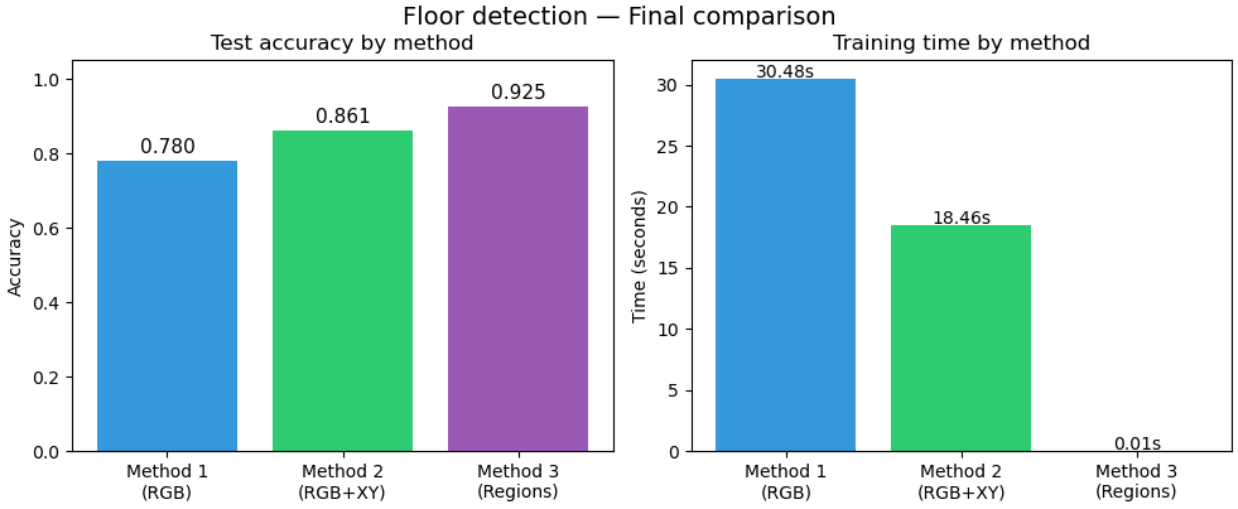
4.1 Performance Comparison

All three methods were evaluated using an 80/20 stratified train-test split with the same random seed to ensure fair comparison. The table below summarizes the key performance metrics:

Method	Test Accuracy	Training Time
Method 1: RGB Only	~78.0%	~30.5 seconds
Method 2: RGB + (x, y)	~86.1%	~18.5 seconds
Method 3: KMeans Regions	**~92.5%**	**~0.008 seconds**

4.2 Key Findings

- **Highest Accuracy:** Method 3 (KMeans region-level) achieved the best test accuracy at 92.5%, demonstrating that region-based approaches effectively capture floor characteristics
- **Fastest Training:** Method 3 trained in just 0.008 seconds, approximately 2,300 times faster than Method 1 and 3,800 times faster than Method 2, due to the dramatic reduction in training samples
- **Spatial Features Matter:** Method 2 improved accuracy by 8.1 percentage points over Method 1, confirming that position information is valuable for floor detection
- **Region-Level Superiority:** The region-based approach outperformed both pixel-level methods while being orders of magnitude faster, making it the clear winner for practical applications



4.3 Evaluation Metrics

Beyond accuracy, the implementation included confusion matrices and classification reports (precision, recall, F1-score) for each method. Visual comparisons included ground-truth masks, predicted masks, and overlay visualizations for sample images, along with side-by-side comparisons of predictions from all three methods on the same images.

5. Discussion

5.1 Why Method 3 Outperforms

The superior performance of the KMeans region-level approach can be attributed to several factors:

- **Noise Reduction:** By averaging pixels within regions, the method reduces the impact of individual noisy pixels or lighting variations
- **Semantic Coherence:** Regions naturally correspond to meaningful image segments, aligning better with how humans perceive floors as continuous surfaces
- **Efficient Representation:** Using 1,200 representative region samples instead of 60,000+ pixel samples allows the SVM to learn more robust decision boundaries without overfitting
- **Combined Features:** Region-level features naturally incorporate both color statistics and spatial position, capturing the essence of what makes a region "floor-like"

5.2 Trade-offs and Considerations

While Method 3 demonstrates clear advantages, some considerations should be noted:

- **Clustering Overhead:** The KMeans clustering step adds preprocessing time, though this is offset by dramatically faster SVM training
- **Hyperparameter Sensitivity:** The choice of 100 clusters affects performance; too few may oversimplify the scene, while too many may reintroduce noise
- **Prediction Granularity:** Predictions are constrained to region boundaries, which may be less precise than pixel-level predictions at object edges

5.3 Real-World Applications

The findings have practical implications for deploying floor detection systems:

- **Resource-Constrained Devices:** Method 3's speed makes it suitable for embedded systems or mobile robots with limited computational power
- **Real-Time Processing:** The 0.008-second training time suggests rapid adaptation to new environments is feasible
- **Scalability:** The region-based approach scales better to larger datasets and higher-resolution images than pixel-level methods

6. Challenges and Future Work

6.1 Challenges Encountered

- **Class Imbalance:** Floor regions often constitute a minority of total pixels, requiring balanced class weights in the SVM
- **Illumination Variation:** Different lighting conditions affect color consistency, potentially confusing color-based classifiers
- **Texture Complexity:** Floors with intricate patterns or multiple materials present feature extraction challenges
- **Limited Dataset Size:** With only 12 images, generalization to diverse indoor environments may be limited

6.2 Proposed Improvements

- **Texture Features:** Incorporate Local Binary Patterns (LBP) or Histogram of Oriented Gradients (HOG) to capture floor textures beyond simple color
- **Multi-Scale Analysis:** Apply region-based classification at multiple KMeans cluster counts to capture both fine details and broad patterns
- **Alternative Kernels:** Experiment with polynomial and sigmoid kernels to compare against the RBF kernel used in this study
- **Deep Learning Comparison:** Benchmark results against modern semantic segmentation architectures like U-Net or DeepLabv3+
- **Cross-Dataset Validation:** Test on publicly available indoor scene datasets to assess generalization capability

7. Conclusion

This study successfully demonstrated three distinct approaches to floor detection using Support Vector Machines, with the KMeans region-level method emerging as the clear winner in both accuracy and computational efficiency. Achieving 92.5% test accuracy with training times of just 0.008 seconds, Method 3 proves that thoughtful feature engineering can yield results competitive with more complex approaches while maintaining practical deployability.

The progression from RGB-only (78.0%) to RGB+spatial (86.1%) to region-based (92.5%) classification illustrates the importance of incorporating both spatial context and semantic grouping. This work confirms that for certain computer vision tasks, classical machine learning methods combined with intelligent feature engineering can provide efficient, interpretable, and effective solutions.

The insights gained from this comparative study provide a foundation for future work in indoor scene understanding and can guide the development of practical floor detection systems for robotics, navigation aids, and augmented reality applications.

8. References

- Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273-297.
- Lloyd, S. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2), 129-137.
- Scikit-learn: Machine Learning in Python. Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
- Lin, T. Y., et al. (2014). Microsoft COCO: Common objects in context. European Conference on Computer Vision.

Appendix A: Implementation Details

A.1 Software and Libraries

- Python: 3.11
- NumPy: For numerical computations
- OpenCV (cv2): Image processing and manipulation
- Matplotlib: Visualization of results
- Scikit-image: Advanced image processing
- Scikit-learn: Machine learning algorithms (SVM, KMeans, StandardScaler)
- PIL (Pillow): Image file handling

A.2 SVM Hyperparameters

- Kernel: RBF (Radial Basis Function)
- C (Regularization parameter): 10
- Gamma: 'scale' (automatically computed as 1 / (n_features × X.var()))
- Class weight: 'balanced' (to handle class imbalance)

A.3 Data Split Configuration

- Train-test split: 80/20
- Stratification: Enabled (to maintain class distribution)
- Random seed: Fixed for reproducibility

A.4 Project Repository

The complete implementation, including the Jupyter notebook (Floor_detection.ipynb), dataset annotations (CMM_Annotations.json), and documentation, is available at:

https://github.com/HackHeroic/Floor_Detection_Svm

https://drive.google.com/drive/folders/1XpAznqPkDR-6IcoDFB7DMxU5vJP1wwKZ?usp=sharing

Acknowledgments

I would like to express my gratitude to the faculty of Newton School of Technology, Rishihood University, for their guidance throughout this Advanced Machine Learning course. Special thanks to my instructor Sanchit sir for providing valuable insights into feature engineering and classical machine learning approaches. This project has significantly enhanced my understanding of the importance of thoughtful feature design in machine learning applications.

--- End of Report ---