Exploring Lightweight Data-Driven Methods for Image Segmentation in the RoboCup SPL

Franziska-Sophie Göttsch[1,2]

[1] Hamburg University of Technology, Hamburg, Germany
[2] HULKs e.V., RoboCup SPL Team Hamburg, Germany hulks@tuhh.de
https://hulks.de

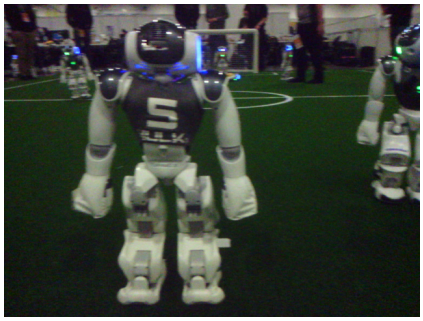
Abstract. Image segmentation is essential in robotics for tasks like self-localization and object detection. In the RoboCup Standard Platform League (SPL), where humanoid Nao robots play soccer autonomously, accurate field segmentation supports key functions but must run on limited hardware, making the design of lightweight yet accurate segmentation methods important.
In this work, we present a comparative study of lightweight segmentation methods in the context of RoboCup SPL. Our focus is on minimizing manual tuning while maximizing accuracy under real-world conditions. We propose an optimization pipeline to efficiently explore several classifiers and features using a zeroth-order optimization framework. The pipeline evaluates combinations of color channels, texture descriptors, and classifiers with the goal of achieving a high F_2 score while keeping computational complexity low.
Our results show that the Nyström approximation of the Radial Basis Function (RBF) kernel achieves the highest accuracy, but is computationally more expensive. In contrast, the Decision Tree (DT) offers a strong balance between accuracy and efficiency, using fewer features and a simpler texture method to achieve reliable performance.

Keywords: Image Segmentation · Data-Driven Methods · RoboCup

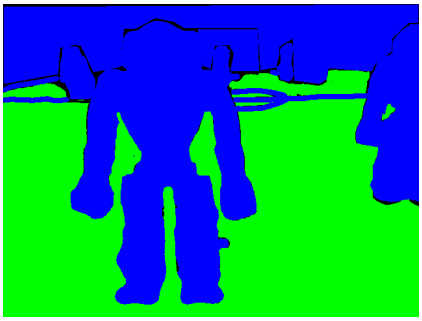
1 Introduction

Image segmentation is a fundamental task in computer vision to classify each pixel of an image into meaningful categories. While recent advances in deep learning have led to highly accurate segmentation methods, such approaches are typically computationally intensive and unsuitable for real-time, resource-constrained systems.

One such example is the Nao robot used in the RoboCup Standard Platform League (SPL), a research competition where university teams program humanoid robots to play soccer autonomously. During a soccer game, the robots must accurately navigate the field and interact with teammates, opponents, and the ball. To perceive their surroundings, Nao robots are equipped with two cameras integrated into the head. These images are commonly used for self-localization



(a) RGB image from Nao's top camera (RoboCup 2019).



(b) Manual segmentation: *field* (green), *not field* (blue).

Fig. 1: Example input image and ground truth segmentation used for training.

and object detection tasks, such as identifying the ball, robots, humans, or goal posts.

To reduce the computational complexity of these perception tasks, one strategy is to pre-process the raw images into segmentation masks that highlight only the relevant visual structures on the field. Key cues in this context include the green artificial turf and the white line markings. A practical simplification is to segment the image into two classes: *field* and *not field*, where the latter one includes lines, robots, the ball, and background elements. An example of such a segmentation is shown in Figure 1.

However, accurately segmenting the field is challenging due to variations in environmental conditions. Lighting may differ significantly in brightness and color temperature. Indoor fields have artificial lighting, but when placed next to windows, they are affected by sunlight. In the latter case, changing daylight conditions casts strong shadows or creates reflections during the match. Additionally, the visibility of line markings may vary depending on whether they are taped or spray-painted. Even the turf itself may have inconsistencies in color or texture. These factors highlight the need for segmentation methods that are robust to environmental variability.

Conventional methods in the RoboCup SPL rely on manually tuned lookup tables or histogram-thresholding on selected color channels [1]. One notable method [6] introduced dynamic adaptation of thresholds based on lighting conditions, improving robustness during games. More advanced techniques, such as fuzzy C-means with Radial Basis Function (RBF) Support Vector Machines (SVMs) [13] exist but were developed for the RoboCup Middle Size League, where greater computational resources are available.

Beyond the RoboCup domain, segmentation methods leverage clustering algorithms, neural networks, or engineered features [10]. However, applying such approaches on the Nao robot requires balancing accuracy with computational efficiency due to strict runtime and hardware constraints.

In this work, we address this trade-off by proposing an optimization pipeline that efficiently explores combinations of color and texture features for various lightweight classifiers. In Section 2, we review common segmentation pipelines and summarize relevant approaches in the RoboCup domain. Section 3 introduces the classifiers and features selected for evaluation. Our optimization strategy is presented in Section 4, followed by a discussion of the experimental results in Section 5, where we highlight the most effective segmentation pipelines for deployment on resource-constrained platforms.

2 Related Work

Image segmentation is a central topic in computer vision with a wide variety of methods that differ in complexity, computational demands, and target applications. Typically, segmentation pipelines involve four stages: image preprocessing, feature extraction, segmentation, and classification [10].

In real-world environments, image pre-processing is essential to mitigate noise, illumination variability, and low contrast. Common techniques include contrast enhancement, normalization, resizing, and denoising, which collectively help improve the quality of subsequent segmentation [10].

Next, feature extraction plays a critical role in segmentation and depends on the domain and input modality. Based on the comprehensive study by Wang et al. [2], features are categorized as morphological, spectral, visual texture, or spatial context features. They highlight that color is a straightforward and natural choice, as color features are directly derived from the input image, leading many approaches to incorporate various color spaces into their feature vectors. Additionally, the selection of specific color channels often depends on the segmentation method employed.

While RGB is the standard format for image display, the coupling of luminance and chrominance often makes it suboptimal for segmentation tasks, particularly under varying lighting conditions. Alternatives, such as HSV, HSI, YCrCb, YUV, L*a*b*, and L*u*v*, have shown improved robustness to lighting variations [2].

Beyond basic color channels, more advanced color descriptors have been proposed. For instance, Excess Green (ExG) indices offer improved segmentation under varying illumination by combining color channels into one index [3]. Looking beyond just the raw color values, gradient-based techniques like the Canny edge detector [13] and Histogram of Oriented Gradients (HoG) [9] capture spatial edge information. Binary descriptors such as Local Binary Patterns (LBPs) or Binary Robust Independent Elementary Features (BRIEF) are simple methods to capture basic texture information [4]. Gabor filters [12] provide additional context by incorporating frequency and pattern-based information. More recently, features learned via deep convolutional neural networks, such as ResNet18, have demonstrated strong performance in segmentation tasks [7].

Next to supervised segmentation methods, such as SVMs or Decision Trees (DTs), unsupervised methods like Fuzzy C-means are also used widely for its simplicity and ability to model overlapping classes [2].

Several works in the RoboCup community have explored handcrafted feature-based segmentation pipelines tailored for the Nao robot system. Basler [1] explores different color spaces for histogram-thresholding. Concluding that the green chromaticity channel g combined with pixel brightness I perform best, with white balancing further enhancing the results. While Qian and Lee [6] similarly applied histogram-thresholding to the green chromaticity channel, they adapt the threshold dynamically in response to lighting changes and head movements, thereby improving segmentation robustness during gameplay. Zhang et al. [13] propose a segmentation approach that combines edge features and statistical color descriptors extracted from multiple color spaces. These features are clustered using Fuzzy C-Means and then classified using a RBF SVM. Their method targets the Middle Size League and assumes access to more powerful computational resources, making it less directly applicable to the real-time, low-resource constraints of the RoboCup SPL.

While numerous segmentation techniques exist, few address the unique constraints of real-time embedded systems. This work fills that gap by proposing a modular evaluation framework for lightweight image segmentation on resource-constrained hardware. We assess combinations of feature extractors and classification methods in terms of accuracy, robustness, and computational cost, using data collected from RoboCup SPL competitions.

3 Explored Methods and Features

In this section, we present the classifiers, color spaces, and texture feature extraction methods that we selected based on their suitability for real-time and resource-constrained systems. These components form the search space explored in our optimization pipeline. Additionally, we briefly describe the Nao robot, focusing on its processing capabilities and camera system, to highlight the hardware limitations that directly influence method selection.

3.1 Classification

We represent the image segmentation as a binary classification problem where each pixel is labeled as either *field* (true/1) or *not field* (false/0). A vector $\mathbf{x}$ from a given feature space $\mathcal{X}$ represents a single pixel with $d \in \mathbb{N}$ features. Each feature vector has a corresponding ground truth label y from a label space $\mathcal{Y}$, which in our case is $\{0,1\}$. A model $M : \mathcal{X} \to \mathcal{Y}$ maps from the feature to the label space and predicts a label $\hat{y} = M(\mathbf{x})$ for a given $\mathbf{x}$.

Histogram-Thresholding: For each feature $i \in \{1, \ldots, d\}$, a closed interval $[t_1^{(i)}, t_2^{(i)}]$ is given, and the collection of all such intervals forms the set T. The classification is then defined by the indicator function $\hat{y} = \mathbf{1}_T(x)$, which returns 1 if $\mathbf{x}$ is within all ranges in T and otherwise 0.

Decision Tree: With a binary tree structure, each internal node compares a feature to a given threshold with a maximum tree height $h \in \mathbb{N}$ leading to the predicted label $\hat{y}$.

Linear Support Vector Machine: For SVMs, classification is based on the sign of a decision function $f(\mathbf{x})$ as shown in the following equation:

$$\hat{y} = \begin{cases} 1 & \text{if } f(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{with } f : \mathcal{X} \to \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \tag{1}$$

Here, $\mathbf{w}$ denotes the weight vector and b the bias term. In the linear case, the decision function takes the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$.

Kernel Method with SVMs: The formulation of linear SVMs above assumes that the data is linearly separable. When this assumption does not hold, the so-called *kernel trick* [8] is applied. Instead of explicitly mapping the data to a higher-dimensional space, a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ computes inner products in a feature space $\mathcal{V}$ induced by the feature map $\phi : \mathcal{X} \to \mathcal{V}$. This removes the need of computing the expensive feature map explicitly and only the kernel function k needs to be evaluated, which is typically more computationally efficient. However, kernel methods suffer from poor scalability in large datasets.

To mitigate this, various kernel approximation techniques, such as the Nyström method [11], approximate the high-dimensional feature map ϕ with a lower-dimensional representation. In the case of the Nyström method, the complexity to compute the kernel function reduces from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$, where $m \ll n$ is the dimensionality of the approximated feature space.

3.2 Color Spaces

Various color spaces are used to represent color information, each offering different advantages depending on the task. The most common ones found in Computer Vision (CV) are RGB, rgbI, HSV, YCrCb, and L*a*b*, where rgbI is the normalized version of the RGB color space, sometimes also referred to as RGB* [2]. In all these color spaces apart from RGB, the intensity is decoupled from the chromaticity. The intensity channels are I, Y, L*, and V. RGB and rgb describe the colors red, green, and blue, respectively. Cb and b* are the blue-yellow axis and Cr and a* the red-green axis. H represents the hue and covers the whole color spectrum and S is the saturation.

3.3 Texture Feature Extraction Methods

While color information is a valuable cue for image segmentation, it is often insufficient when different object classes exhibit similar color values. To overcome this limitation and improve classification accuracy, local texture features are used to provide more context. This section introduces four methods that are commonly found in CV pipelines, representing a diverse range of approaches:

raw spatial sampling, binary encoding, edge orientation histograms, and convolutional filter responses. This selection enables a balanced comparison across simplicity, computational efficiency, and descriptive power.

Texture feature methods are typically defined to be applied to a gray scale image but are also applicable to images with multiple color channels by applying the method to each color channel separately. The following focuses on the application to a single color channel.

Neighboring Pixels: The raw color value of the neighboring pixels around a center pixel are sampled at radius r and with o equidistant orientations. This augments the feature vector by o additional features per pixel, capturing local spatial context.

Local Binary Pattern: This method samples 8 pixel values around a center pixel with radius r. The intensity of each of these pixels is compared to that of the center pixel and assigned either a 0 or 1 if the value is smaller or greater, respectively. The resulting binary pattern is encoded as a single integer, adding exactly one feature per pixel. This compact representation provides robustness to illumination changes while remaining computationally efficient.

Histogram of Oriented Gradients: HoG involves computing gradient magnitudes and orientations within local cells, which are then binned into a fixed number of orientation bins o. The dimensionality of the resulting feature vector depends on the number of bins and the cell configuration. While the per-pixel complexity remains linear, the computational cost is higher than that of LBP due to the additional gradient and histogram calculations [4].

Gabor Filters: The impulse response of a Gabor filter is defined by a sinusoidal wave and a Gaussian function [5] to capture frequency and orientation-specific information:

$$g(x, y; \lambda, \theta, \phi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(\frac{2\pi x'}{\lambda} + \phi\right)\right), \tag{2a}$$

$$x' = x\cos\theta + y\sin\theta, y' = -x\sin\theta + y\cos\theta. \tag{2b}$$

The parameters in the Gabor function play specific roles: σ controls the standard deviation of the Gaussian envelope and thus the spatial extent of the filter. $\theta \in [0, 2\pi]$ defines the orientation of the filter, enabling directional sensitivity. $\lambda \in [0, 2\pi]$ determines the wavelength of the sinusoidal component, which sets the preferred spatial frequency. $\gamma \in [0, 1]$ adjusts the spatial aspect ratio, where smaller values create elongated filters and values closer to 1 approach circular symmetry. And $\phi \in [0, 2\pi]$ sets the phase offset, influencing the symmetry of the filter response. Each filter is then convolved with the image, where the number of filters K, as well as the kernel size k determine the computational cost [10].

3.4 The Nao Robot

The Nao v6 by Aldebaran Robotics is the official standard platform in the SPL and to be used by all participating teams. Released in 2018, the Nao v6 features

relatively modest hardware by today's standards, with an Intel Atom quad-core 1.91 GHz CPU, 4 GB of DDR3 RAM, and a 32 GB SSD. It is equipped with two head-mounted cameras capable of capturing images at various resolutions depending on the selected frame rate. The images are recorded in RGB, but then compressed in a lossy way into YCrCb422 due to hardware limitations.

4 Finding the most efficient Method for Image Segmentation

Based on the methods described in Section 3, we next discuss the optimization and training process to find the best method for lightweight image segmentation. We present our dataset, motivate certain design choices based on the requirements given by the Nao robot and the SPL context, and introduce the proposed optimization pipeline.

4.1 Dataset

Our dataset consist of images captured by the Nao v6 from different teams and events, leading to a vide variety within the images. Depending on the frame rate, the Nao can record in different resolutions. In this work, we only consider a resolution of 640x480 for both the top and bottom camera. The full dataset contains 1,864,394 images from which 1,374,268 are in the wanted resolution. Around 70% of the images with a resolution of 640x480 are from the top camera while the remaining 30% are from the bottom camera. The bottom camera is tilted downwards to only capture objects directly in front of the robot and thus, often only shows field, the robots feet, and the ball. The top camera, on the other hand, is tilted more towards the horizon, also capturing objects further away, like other robots, more lines, or the goalposts. Thus, the segmentation of the top camera images is more complex which is why we choose to include more top camera images in the dataset than bottom camera images.

From this dataset, we manually labeled 601 images. The ground truth is a segmentation mask with three classes: *field*, *not field*, and *uncertain*. The class *not field* is further categorized into lines, robots, goal, and others, e.g., background or humans. The pixels from the class *uncertain* are not used for training or testing. Furthermore, the images are each tagged based on their lighting, field, and line conditions as shown in Figure 2. Based on these tags, we choose to sample equal amounts of images from indoor, outdoor, and mixed scenarios. And within these three scenarios, the appearance of bright light reflections and shadows has been equalized where possible. Examples of images with their categories are shown in figure Figure 3. This categorization allows for more precise training and testing the model under varying conditions.

4.2 Optimization Pipeline

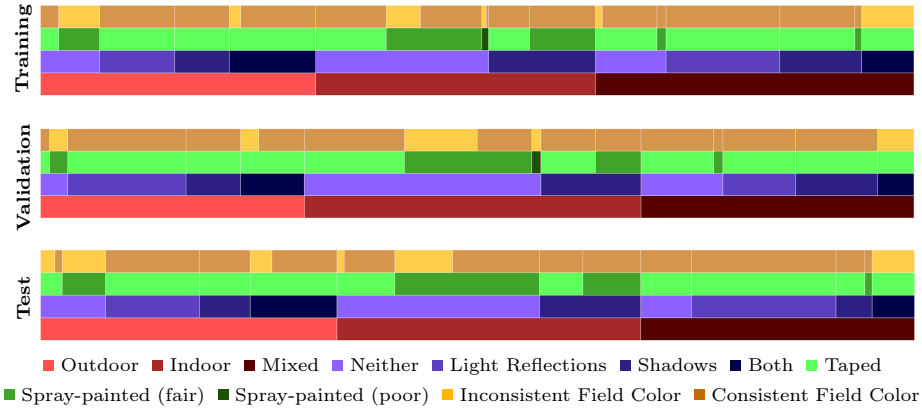
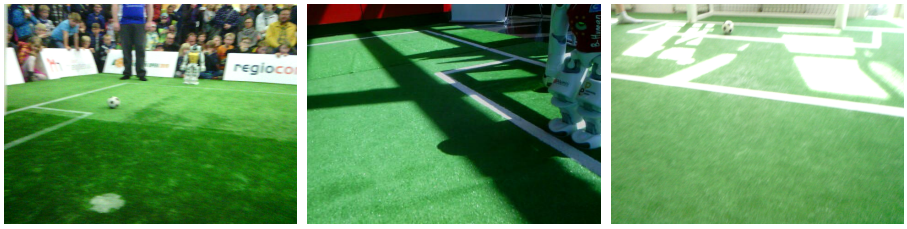


Fig. 2: Distribution of the train (64%), validation (16%), and test (20%) data splits.



(a) Indoor, Neither, Spray-painted (poor), Inconsistent Field Color.

(b) Outdoor, Shadows, Taped, Consistent Field Color.

(c) Outdoor, Light Reflections, Taped, Consistent Field Color.

Fig. 3: Examples of images from the dataset with their corresponding categories.

The optimization and training pipeline is implemented in Python due to its rich ecosystem of libraries for Machine Learning (ML) and CV. We mostly rely on *OpenCV* and *scikit-image* for color space transformations and texture feature extraction, *scikit-learn* for classifier training, and *Optuna* for hyperparameter optimization.

Initially, only the training and validation dataset is used. For each classifier, an independent study is initialized to explore the respective hyperparameter space using a Tree-structured Parzen Estimator (TPE) sampler. This pipeline is illustrated in Figure 4.

Each optimization trial consists of several steps. First, the sampler selects the parameters for the current trial. This includes a non-empty subset of color channels from the full set of color channels. One of these selected channels is used for the texture feature extraction. A texture feature extraction method is chosen along with its corresponding parameters (see Section 3.3). The texture and color features are extracted according to the sampled parameters and form

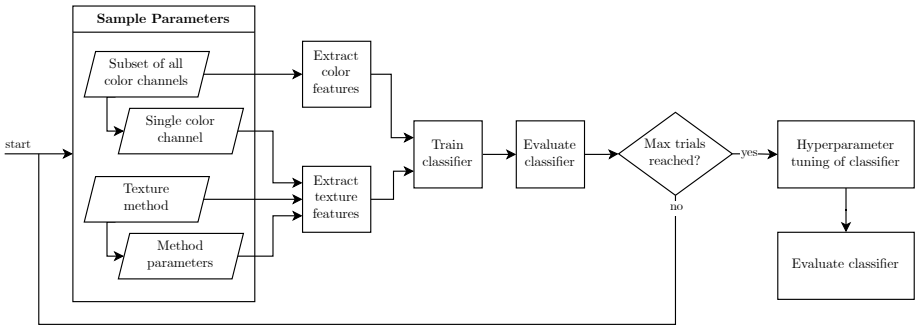


Fig. 4: Proposed optimization pipeline.

the feature vectors for each sample. Based on this representation, a classifier is trained on the training dataset and evaluated on the validation dataset. Based on this scoring and the number of selected color channels, the TPE sampler suggests a new configuration to improve the performance. This iterative process continues until a predefined number of trials are completed.

After identifying the most effective combination of color channels and texture features for a given classifier, a final tuning phase is performed. In this phase, classifier-specific hyperparameters are optimized using a grid search with five-fold cross-validation on the combined training and validation set. The final score of each classifier is obtained by applying it to the test dataset.

As discussed in Section 2, typical segmentation pipelines include pre-processing or post-processing steps to enhance the quality of the segmentation mask. While these steps are effective, they typically require additional computational resources. Due to the limited processing power of the Nao robot, such operations would violate the real-time constraints. Therefore, we restrict our pipeline to use only the raw input images and the direct output of the classifier, without any additional image processing.

4.3 Performance Metrics

To evaluate the performance of the classifier at the end of each optimization trial, we use the F_β score in Equation (3) with $\beta = 2$:

$$\mathrm{F}_\beta = \frac{\beta^2 + 1}{\beta^2 \cdot \mathrm{recall}^{-1} + \mathrm{precision}^{-1}} = \frac{(1+\beta^2) \cdot \mathrm{TP}}{(1+\beta^2) \cdot \mathrm{TP} + \beta^2 \cdot \mathrm{FN} + \mathrm{FP}}. \tag{3}$$

This setting prioritizes recall over precision, i.e. false negatives will be penalized more than false positives. This decision is motivated by the usage of the segmented images in the line detection of the robot. The line detection algorithm tries to fit lines through the pixels classified as *not field* and thus, false negatives, i.e., pixels that are *field* but classified as *not field*, significantly degrade first line

detection and consequently also localization. This F_2 score is used by the TPE sampler and shall be maximized.

In addition to the F_2 score, the TPE sampler also uses the number of selected color channels as an objective. This number shall be minimized to reduce the need of color space transformation which are computationally expensive, especially on the Nao.

5 Evaluation

This section first introduces the baseline to which we compare the tuned classifiers. Next, we present the results of the optimization pipeline described in Section 4.2. Here, each classifier's performance is analyzed with respect to two criteria, segmentation accuracy and its overall computational complexity.

5.1 Histogram-Thresholding as Baseline

To show the influence of the data-driven classifier and the texture feature in addition to the color features, we chose simple histogram-thresholding as a baseline. The parameters, i.e., the selected color channels and ranges, were tuned at the RoboCup 2024 for the specific location. The optimal parameters may vary from location to location but since we are looking for a solution that works in all scenarios, we use these parameters on the whole dataset. The results of the baseline classifier are included in Table 1.

5.2 Overall Performance during Optimization

The Pareto fronts, defined by the two optimization targets, in Figure 5 illustrate how the TPE sampler explored the trade-off between classification performance (F_2 score) and the number of selected color channels. For all classifiers, a clear Pareto front emerged, with most recent trials clustering near the front. This suggests convergence toward optimal solutions, even though a TPE sampling-based, zeroth-order optimizer cannot guarantee globally optimal results. For each classifier, we ran the optimization pipeline for 2000 trials. Primarily due to time constraints, but also because the TPE sampler did not find significantly better parameters beyond this point.

Notably, the best-performing trials for DT and both kernel approximation methods (Polynomial and RBF) used only two or three color channels. As more channels are added, performance decreased, particularly for the kernel approximations. This may be due to the fixed number of components used in the approximated high-dimensional space. More input features mean that the feature vector cannot be expanded as much as one with fewer dimensions, weakening the kernel approximation's effectiveness.

The Polynomial kernel approximation showed high variance in F_2 scores. Some trials performed well, while others dropped significantly below 80%. In contrast, the RBF approximation is unusually stable, rarely scoring below 90%.

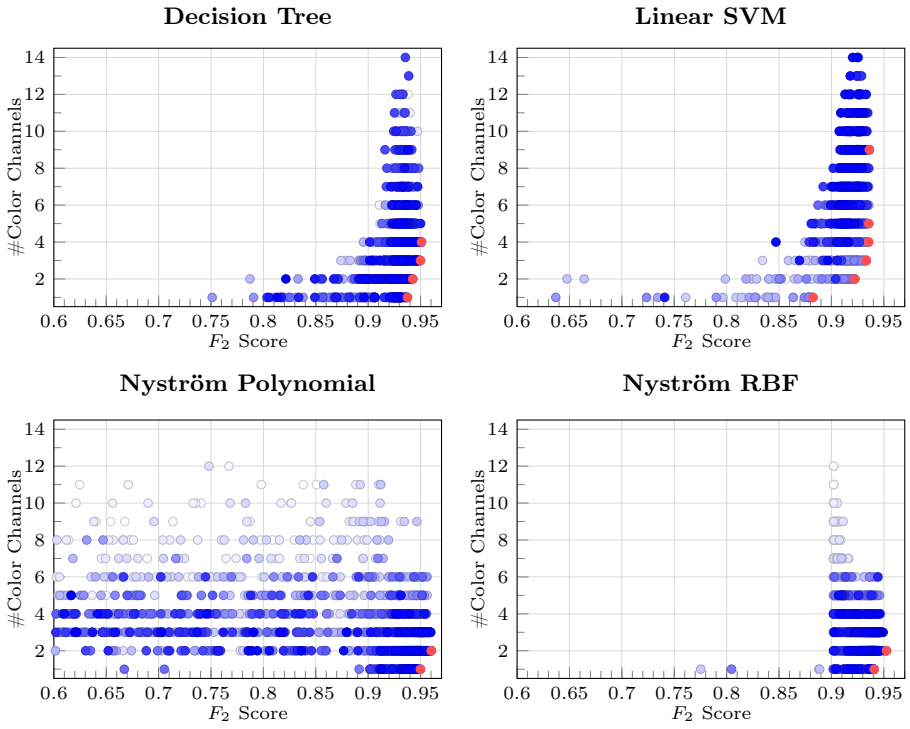


Fig. 5: The Pareto fronts of each classifier with the trial number and best trials found by a TPE sampler during the optimization.

Despite this, the top trials from both kernel methods performe comparably or better than the other classifiers. The high variance in the Polynomial kernel's performance suggests that the feature space does not consistently support the complex decision boundaries it models. In contrast, the RBF kernel's stable results suggests that the feature space is better suited to these types of decision boundaries.

The linear SVM struggled when few color channels are selected and plateaued in performance beyond five channels, never matching the best scores of the other methods. This makes the linear SVM less suitable for our use case since each color channel requires a transformation and thus, increases the computational load on the Nao. And since selecting more color channels does not improve the accuracy, other methods are preferred even though linear SVM is a lightweight method.

Analysis of the best trials in Table 1 reveals a preference for certain texture methods. With DT and both kernel approximations, the best trials have the simple texture methods Neighboring Pixel (NP) and LBP, while the linear SVM has the more complex HoG. The choice of an expensive texture feature method is one more reason why linear SVM is less suitable. Additionally, the

best-performing pipelines consistently used a single dominant color channel for texture extraction, typically the L* or g channel. Interestingly, only the linear SVM preferred a luminance-based channel while the others favored chromaticity features.

Group	ID	F2 Score	F2 Score (tuned)	Color Channels	Texture Method
Baseline	-	0.8745	-	Y, G, H, S	-
DT	1577	0.9509	0.9557	Y, **Cb**, B, I	NP
	649	0.9499	0.9621	Y, **Cb**, B	NP
	1222	0.9399	0.9519	**Cr**, g	NP
	1543	0.9374	0.9656	**g**	NP
Linear SVM	614	0.9357	0.9545	Cr, B, r, **L***, a*	HoG
	735	0.9356	0.9524	Cb, R, **L***, a*	HoG
	844	0.9332	0.9537	R, **L***, a*	HoG
	952	0.9221	0.9487	**L***, a*	HoG
	280	0.8825	0.9175	**g**	HoG
Nyström RBF	1067	0.9526	0.9678	**g**, a*	LBP
	1020	0.9408	0.9470	**a***	LBP
Nyström Polynomial	1626	0.9357	0.9619	**g**, S	NP
	1940	0.9356	0.9554	**g**	NP

Table 1: Results of the best trials. The channels marked in bold are used for the texture feature extraction.

While trends are clearer within classifiers, some consistency appears across the different groups. For example, the green chromaticity channel frequently emerged in optimal configurations, which aligns with previous findings in [1]. Furthermore, with only one or two color channels selected, chromaticity channels such as g, a*, Cr, H, and S are preferred. From three color channels and up, at least one intensity channel like Y, L*, or I is selected. This suggests that, for our use case, the chromaticity channels are more relevant, especially for the texture features. Furthermore, the histograms over the color values in Figure 6 show that the most frequently selected color channels, such as Y, g, a*, and S exhibit a high degree of class separability. In contrast, L* is selected in all best trials of the linear SVM despite showing less evident separation between classes. This suggests that L* may contribute to a linearly separable feature space when combined with one or more other color channels, making it suitable for the linear SVM. Lastly, although the G channel is used in the baseline, it is not selected in any of the best trials and also shows limited class separation in the histogram. While the baseline thresholds may have performed well on the specific game data they were derived from, they do not generalize well to the full dataset, which includes more diverse lighting and field conditions.

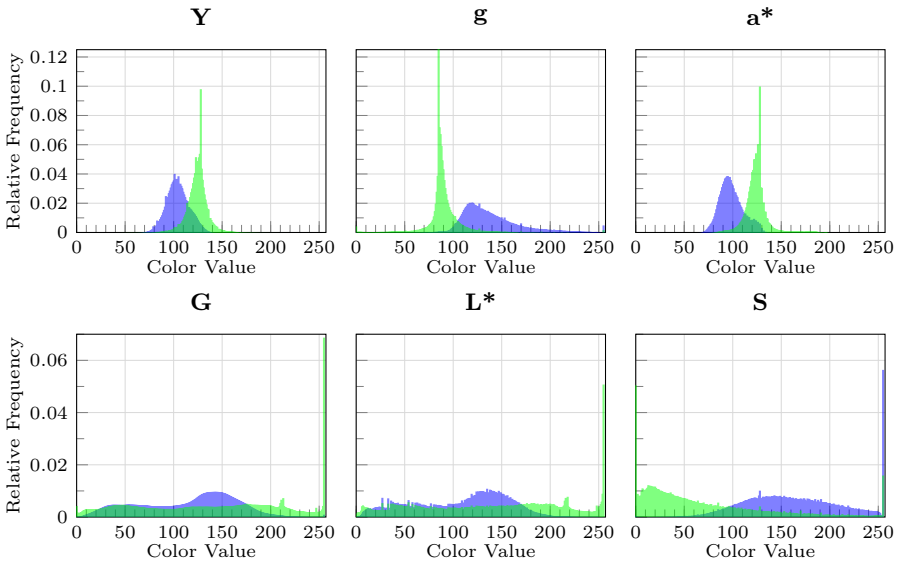


Fig. 6: Histograms of selected color channels showing the distribution over the color values for the two classes *field* ■ and *not field* ■.

5.3 Classifier-Specific Hyperparameter Tuning

As expected, all classifiers improved slightly after dedicated hyperparameter tuning, with F_2 scores increasing by approximately 1 percentage point across the board (see F_2 *Score (tuned)* in Table 1). These results confirm the added value of classifier-specific fine-tuning in the second optimization stage.

Both kernel approximation methods increased the number of components from 50 to 100, effectively enhancing the quality of the approximated kernel space. For DT, a tree depth of five has the best results which shows that even a small tree achieves high accuracy. The best DT only takes one color feature and the color values from the sampled pixels in the neighborhood. Thus, the feature vector is already small, reducing the need for a deep DT.

5.4 Final Results on the Test Dataset

Applying the tuned models to the test dataset reveals more nuanced strengths and weaknesses beyond what the F_2 score alone suggests. Some examples of predicted segmentation masks are shown in Figure 7. The baseline histogram-thresholding method performs well under uniform lighting (see bottom row) but struggles with extreme brightness or darkness. Here, common misclassifications in the form of false positives are robots, the black spots on the ball, lines with shadows, and background. False negatives often appear with dark lighting (see second row).

The DT model significantly reduces the false positives, especially in the background and line regions with shadows. However, the classifier tends to underperform in darker scenes leading to higher false negatives. The increase in false negatives is unfavorable since a line detection algorithm could possibly fit a line through the noise at the bottom which affects the self-localization of the robot. The linear SVM performs similarly to the DT but incurs more false negatives, which limits its utility for scenarios like ours where recall is critical.

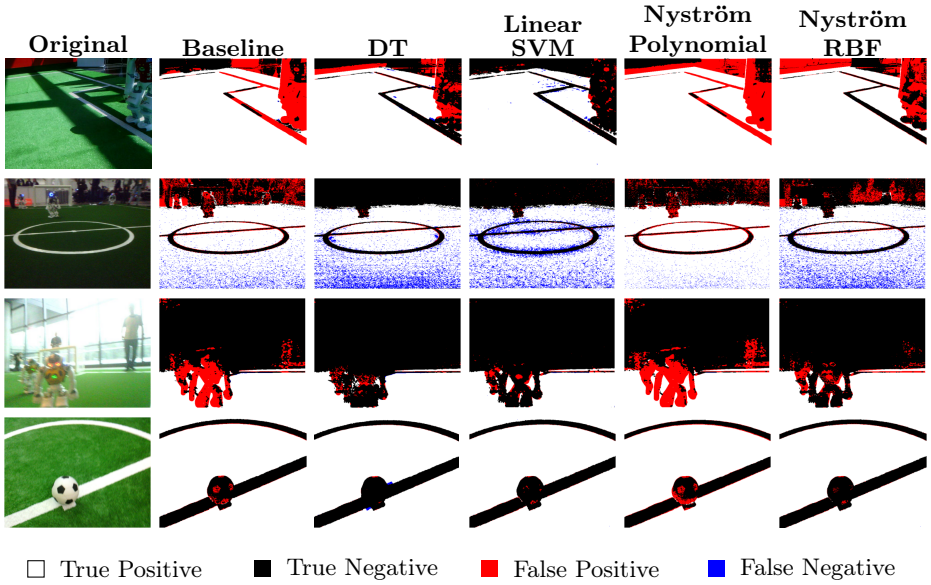


Fig. 7: Predicted segmentation masks with confusion matrix of the baseline and the best classifiers

The Polynomial kernel approximation excels in detecting field regions under low light and has the fewest false negatives. However, it suffers from frequent false positives and misclassifying faint lines, robots, and background elements. Even thought the F_2 score is similar to the other methods, the bias towards fewer false negatives hides the fact that the false positive rate is very high, making this method unsuitable for our use case.

The RBF kernel approximation achieved the highest F_2 score overall. It showed fewer false positives than the baseline, particularly in distinguishing robots and background, but still struggled with faint line markings under challenging lighting. The false negative rate is similar to the one of the baseline.

Overall, the DT, linear SVM, and RBF kernel approximation are able to reduce the number of false positives. However, all methods, apart from Polynomial kernel approximation, underperform in darker scenes. A possible solution to this problem would be to use the categorized dataset and split it, e.g., into darker

(most commonly indoor scenes) and brighter scenes, to train multiple variants of a classifier for varying lighting conditions. Before a game, one would then choose the most suitable classifier which could already increase the accuracy without any additional computational resources needed.

Additionally, since the baseline performs well under good lighting conditions, a two-stage classification pipeline could help reduce computational load. In this approach, a fast, color-based method, such as a DT or SVM, would first attempt the classification. Only if the confidence is low would a more complex, texture-augmented classifier be used. This setup balances the trade-off between performance and efficiency: always using the most accurate method (with both color and texture features) yields the best results but is computationally expensive. Simpler classifiers or lightweight features save resources but may be less accurate.

A two-stage approach offers a compromise. Cheap, fast methods handle easy cases, while more demanding methods step in only when needed. This makes the system more efficient and suitable for real-time, resource-limited platforms like the Nao robot. However, this strategy has not yet been implemented.

Overall, no single method clearly outperforms the others. The RBF kernel approximation achieves the highest F_2 score and produces segmentation masks with fewer false positives and no increase in false negatives, making it the most accurate approach. However, this accuracy comes at a cost, the method is computationally expensive, and it remains uncertain whether it can run efficiently on the Nao robot.

In contrast, the DT further reduces false positives but introduces more false negatives. Still, it is significantly more lightweight in terms of computation, using fewer color channels and a simpler texture feature method. Because of its strong balance between accuracy and efficiency, the DT shows the most promise for our image segmentation task, particularly when using multiple decision trees tailored for different lighting conditions or integrating it into a two-stage classification pipeline.

6 Conclusion and Future Work

This paper presented an effective optimization pipeline for developing lightweight and accurate segmentation methods tailored to the constraints of the Nao robot. By systematically combining different color and texture features with various classifiers, we are able to obtain models that achieve high F_2 scores while keeping computational demands low.

Among the tested methods, the DT and the RBF kernel approximation offered the best trade-offs between accuracy and efficiency. The DT stood out as the most practical choice for real-time use, offering solid performance with minimal resource requirements thanks to its simplicity and the use of the inexpensive Neighboring Pixels texture descriptor. Its accuracy and practicability can be further enhanced by training separate models for different scenarios or

by employing a two-stage classification pipeline, starting with a fast, low-cost method and falling back on a more complex one only when necessary.

Overall, this work demonstrates significant progress toward building domain-specific, low-computation segmentation pipelines for embedded systems like the Nao robots used in the RoboCup SPL. Future research may explore additional classifiers such as Multi-Layer Perceptrons, expand the range of texture descriptors, or develop ensemble or condition-aware models, e.g., ones tuned for different lighting environments. Ultimately, the most promising pipeline will be deployed and tested directly on the Nao robot to evaluate its real-time performance during actual match conditions.

References

1. Basler, J.F.: Field Color Detection using Illumination Invariant Imaging in the RoboCup Standard Platform League
2. Garcia-Lamont, F., Cervantes, J., López, A., Rodriguez, L.: Segmentation of images by color features: A survey. Neurocomputing **292**, 1–27 (May 2018). https://doi.org/10.1016/j.neucom.2018.01.091, https://www.sciencedirect.com/science/article/pii/S0925231218302364
3. Guo, W., Rage, U.K., Ninomiya, S.: Illumination invariant segmentation of vegetation for time series wheat images based on decision tree model. Computers and Electronics in Agriculture **96**, 58–66 (Aug 2013). https://doi.org/10.1016/j.compag.2013.04.010, https://www.sciencedirect.com/science/article/pii/S0168169913000847
4. Joshi, K., Patel, M.I.: Recent advances in local feature detector and descriptor: a literature survey. International Journal of Multimedia Information Retrieval **9**(4), 231–247 (Dec 2020). https://doi.org/10.1007/s13735-020-00200-3, https://doi.org/10.1007/s13735-020-00200-3
5. Khan, J.F., Adhami, R.R., Bhuiyan, S.M.A.: A customized Gabor filter for unsupervised color image segmentation. Image and Vision Computing **27**(4), 489–501 (Mar 2009). https://doi.org/10.1016/j.imavis.2008.07.001, https://www.sciencedirect.com/science/article/pii/S0262885608001480
6. Qian, Y., Lee, D.D.: Adaptive Field Detection and Localization in Robot Soccer. In: Behnke, S., Sheh, R., Sarıel, S., Lee, D.D. (eds.) RoboCup 2016: Robot World Cup XX. pp. 218–229. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-68792-6_18
7. Sahu, Y., Tripathi, A., Gupta, R.K., Gautam, P., Pateriya, R.K., Gupta, A.: A CNN-SVM based computer aided diagnosis of breast Cancer using histogram K-means segmentation technique. Multimedia Tools and Applications **82**(9), 14055–14075 (Apr 2023). https://doi.org/10.1007/s11042-022-13807-x, https://doi.org/10.1007/s11042-022-13807-x
8. Schölkopf, B.: The Kernel Trick for Distances. In: Advances in Neural Information Processing Systems. vol. 13. MIT Press (2000), https://proceedings.neurips.cc/paper/2000/hash/4e87337f366f72daa424dae11df0538c-Abstract.html
9. Seeland, M., Rzanny, M., Alaqraa, N., Wäldchen, J., Mäder, P.: Plant species classification using flower images—A comparative study of local feature representations. PLOS ONE **12**(2),

e0170629 (Feb 2017). https://doi.org/10.1371/journal.pone.0170629, https://dx.plos.org/10.1371/journal.pone.0170629
10. Wang, A., Zhang, W., Wei, X.: A review on weed detection using ground-based machine vision and image processing techniques. Computers and Electronics in Agriculture **158**, 226–240 (Mar 2019). https://doi.org/10.1016/j.compag.2019.02.005, https://www.sciencedirect.com/science/article/pii/S0168169918317150
11. Williams, C., Seeger, M.: Using the nyström method to speed up kernel machines. In: Advances in Neural Information Processing Systems. vol. 13. MIT Press (2000)
12. Yang, H.Y., Wang, X.Y., Wang, Q.Y., Zhang, X.J.: LS-SVM based image segmentation using color and texture information. Journal of Visual Communication and Image Representation **23**(7), 1095–1112 (Oct 2012). https://doi.org/10.1016/j.jvcir.2012.07.007, https://www.sciencedirect.com/science/article/pii/S1047320312001241
13. Zhang, R., Chen, G., Wang, Z., Chi, W., Wang, Z., Sun, L., Yang, G., Wen, Y.: Multi-color space learning for image segmentation based on a support vector machine. OSA Continuum **2**(11), 3050–3065 (Nov 2019). https://doi.org/10.1364/OSAC.2.003050, https://opg.optica.org/osac/abstract.cfm?uri=osac-2-11-3050, publisher: Optica Publishing Group