Evaluation of Color Spaces for Robust Image Segmentation

Alexander Jungmann, Jan Jatzkowski and Bernd Kleinjohann
Cooperative Computing & Communication Laboratory
University of Paderborn, Fuerstenallee 11, Paderborn, Germany

Keywords: Image Processing, Color-based Segmentation, Color Spaces, Evaluation of Segmentation Results.

Abstract: In this paper, we evaluate the robustness of our color-based segmentation approach in combination with different color spaces, namely RGB, L*a*b*, HSV, and log-chromaticity (LCCS). For this purpose, we describe our deterministic segmentation algorithm including its gradually transformation of pixel-precise image data into a less error-prone and therefore more robust statistical representation in terms of moments. To investigate the robustness of a specific segmentation setting, we introduce our evaluation framework that directly works on the statistical representation. It is based on two different types of robustness measures, namely relative and absolute robustness. While relative robustness measures stability of segmentation results over time, absolute robustness measures stability regarding varying illumination by comparing results with ground truth data. The significance of these robustness measures is shown by evaluating our segmentation approach with different color spaces. For the evaluation process, an artificial scene was chosen as representative for application scenarios based on artificial landmarks.

1 INTRODUCTION

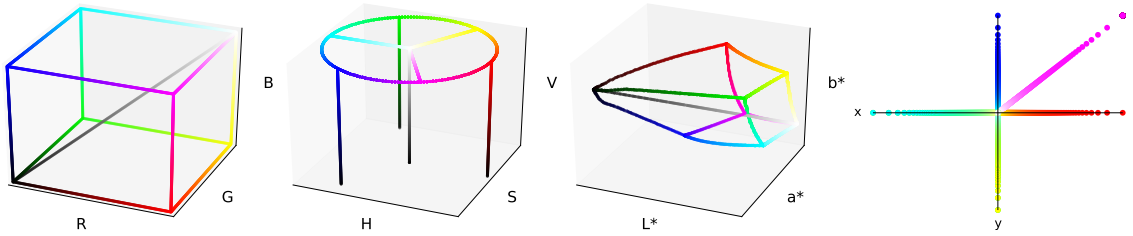
Image segmentation refers to the problem of partitioning a single image into regions of interest (ROI) and a variety of techniques exists for region-based segmentation (Freixenet et al., 2002). Each ROI describes a homogenous region where homogeneity is measured, e.g., by means of features as color, texture, or contours (Russell and Norvig, 2010). Which feature or combination of features is used to measure homogeneity depends on the goal of a certain computer vision system, i.e. there is an ambiguous choice. Usually, a computer vision system includes object detection and recognition whose results are used for high level applications. Features chosen for segmentation depend on object characteristics that are assumed to distinguish objects from their environment. Examples from embedded systems domain are, e.g., driver assistance systems for traffic sign recognition in the automotive domain (Mogelmose et al., 2012), obstacle detection for behavior in robotics (Blas et al., 2008), or face recognition in biometrics (Yang et al., 2009).

In this paper, we focus on color-based segmentation for an efficient and robust object detection within embedded systems. While efficiency is related to the implementation of our segmentation approach, robustness refers to stable segmentation results in presence of illumination variations and noise over time. Since environmental illumination influences color perception, an evaluation of different color spaces seems promising to achieve robustness.

Various color spaces exist, each aiming at providing special characteristics, e.g., regarding color distance, color description, or color ordering. While some color spaces are based on the assumption that each color is composed of three primary colors (trichromatic theory; e.g. RGB, XYZ), other color spaces aim at adapting human color perception (e.g. HSV). Since color-based segmentation utilizes color distances to measure homogeneity of image partitions, the choice of color space has heavy impact on segmentation results (Busin et al., 2009).

In this paper, we propose an efficient, color-based segmentation approach. Furthermore, we present an evaluation of this approach considering varying illumination and different color spaces, namely RGB, L*a*b*, HSV, and LCCS. The goal of this evaluation is to identify color spaces that provide robust results with our segmentation approach. Due to the ill-posed nature of the segmentation problem, evaluation of segmentation results is a hard challenge (Unnikrishnan et al., 2007). On the one hand, results of an entire vision system including segmentation typically contain effects of post-processing steps and therefore do not allow to identify segmentation effects. On the other hand, comparing segmentation results

Jungmann A., Jatzkowski J. and Kleinjohann B..
Evaluation of Color Spaces for Robust Image Segmentation.
DOI: 10.5220/0004743406480655
In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (VISAPP-2014), pages 648-655
ISBN: 978-989-758-003-1



(a) RGB cube edges. (b) HSV color space. (c) L*a*b* color space. (d) Log-chromaticity color space.

Figure 1: Visualization of colors from RGB cube edges within considered color spaces.

with ground truth data raises the question: What is a ground truth segmentation? Considering these challenges of evaluating segmentation results, we propose a relative and an absolute robustness measure within our evaluation framework. By means of these values, we aim at measuring robustness of segmentation results within a still scene sequence. In this context, robustness refers not only to constant regions over time but also to constant regions at different illumination.

2 COLOR SPACES

Color spaces allow to describe a particular color by means of basic components. The variety of color spaces results from differences within these basic components and many color spaces result from each other by transformation. Hence, different color spaces do not have to vary in their covered colors, but rather differ in the order and scaling of colors. Depending on the color space, this results in different distances and distance relations between colors and therefore effects color-based segmentation (cf. Section 5.1).

In this paper, we consider representatives of primary, luminance-chrominance, and perceptual color space classes introduced in (Vandenbroucke et al., 2003) as well as a variant of log-chromaticity color space (LCCS) presented in (Finlayson et al., 2009).

RGB and XYZ are well known representatives of primary color spaces. Since XYZ results from a linear transformation of RGB and RGB is most widely used in color image segmentation (Busin et al., 2009), we consider RGB. In RGB, color is described by its percentage of the primary colors red, green, and blue. Due to equidistant scaling of all three primary colors, RGB color space describes a cube where the main diagonal represents gray values (Figure 1(a)).

The perceptual color space HSV describes a color in cylinder coordinates by its hue, saturation, and a luminance-inspired value (Szeliski, 2011). Hue is described by the angle around the vertical axis with complementary colors 180° opposite one another; saturation and value are usually ranged in $[0,1]$ where saturation $S = 0$ corresponds to shades of gray (Figure 1(b)). In fact, HSV and RGB are related to each other by a non-linear transformation.

The luminance-chrominance color space L*a*b* (also called CIELAB) was originally constructed by International Commission on Illumination (CIE) with a non-linear remapping of the primary XYZ color space to describe differences of colors w.r.t. luminance and chrominance more perceptually uniform (Szeliski, 2011). While a* describes green and red percentage of a color and b* describes its blue and yellow percentage, L* represents luminance. Hence, L*-axis represents all levels of gray (Figure 1(c)).

Furthermore, we consider LCCS which results from a logarithmic scaled projection of RGB color space. In (Finlayson et al., 2009), LCCS is derived based on assumptions of Lambertian reflectance, Planckian lighting, and fairly narrow-band camera sensors. In our evaluation we compute log-chromaticity colors according to (Khanal et al., 2012):

$$(x,y) = (\log(R/G), \log(B/G)). \qquad (1)$$

Figure 1(d) shows a visualization of RGB cube edges transformed to LCCS by Equation 1. Since projection effects that one LCCS color represents various RGB colors, e.g., the point of origin represents all gray values, this visualization is ambiguous.

3 IMAGE SEGMENTATION

The basic idea of image segmentation is to identify contiguous blocks of pixels that are homogeneous with respect to a pre-defined criterion. In our work, we incorporate color as criterion of homogeneity.

3.1 Segmentation Algorithm

Our deterministic segmentation algorithm processes an entire image row by row, starting at the topmost row, at the leftmost pixel within each row. Contiguous blocks of pixels with similar color are identified and immediately extracted from the image. The algorithm incorporates two major concepts. (1) Within a

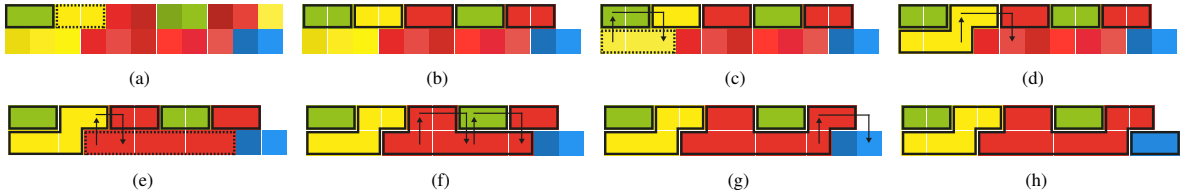


Figure 2: Segmentation example with exemplary intermediate steps. (a) Identified one region (solid) and one run (dashed) in first row. (b) Identified five regions in first row. (c) Region growing step in second row. (d) Finished region growing step in second row. (e) Started region growing step of second run. (f) Finished region growing step of second run. (g) Finished region merging step. (h) Segmentation result. Five regions were identified, one pixel was skipped.

row, adjacent pixels with similar color are compactly represented as *run* by means of *run-length encoding*. (2) Across adjacent rows, a deterministic *region growing* and *merging* approach aggregates sets of runs with similar color. Similarity of color depends on the underlying color space and is measured by color space specific heuristics (cf. Section 5.1).

3.1.1 Run Construction

A block of adjacent pixels with similar color values within a row is compactly represented as run:

$$run_i = \langle (x_1, y_1)_i, l_i \rangle , \tag{2}$$

with $(x_1, y_1)_i$ being the coordinates of the left most associated pixel P_{1_i} and l_i being the block's length. While processing a single row, adjacent pixels with similar color are identified and stored as runs. A new run is started when either a new row begins or after the previous run was finalized. A current run is finalized, when the end of the current row is reached or when the color of the next pixel differs too much from the representative color of the current run.

When starting a new run – let's say run_i – the coordinates of the current pixel are stored to $(x_1, y_1)_i$, length l_i is set to 1 and the color values of the current pixel are saved as representative color values of run_i. The process of adding a new pixel to run_i consists of increasing length l_i by 1 and recalculating the representative color values of run_i accordingly. A constructed run is discarded if its length is smaller than a threshold value l_{min}. After constructing a sufficiently long run and before starting a new run, region growing and region merging takes place.

3.1.2 Region Growing by Aggregating Runs

Adjacent runs within the same row as well as across adjacent rows are aggregated into a region if they have similar color values. A region consists of a set of associated runs:

$$region_j = \bigcup_{i=1}^{n_j} run_i, \tag{3}$$

with n_j being the number of runs associated to $region_j$. Furthermore, each region possesses a representative color that is calculated based on all associated pixels.

A single run is initialized as new region if region growing is not possible (Figure 2(a)) or if no adjacent region with similar color values exists (Figure 2(b)). The region merging step for a run run_i starts at its left most pixel position $(x_1, y_1)_i$ and looks up the region which is associated with the pixel at the same column in the previous row (Figure 2(c)). If such a region – let's say $region_j$ – exists and $region_j$ and run_i have similar color values, run_i is added to $region_j$ by inserting the run into the region's set of runs (Figure 2(d)). The representative color values of $region_j$ are updated accordingly. The region growing process is finished at this point since run_i now belongs to $region_j$ and cannot be added to any further adjacent region in the previous row. Top adjacent regions that have not been considered until now (like, e.g., the green and second red region in Figure 2(f)) all are candidates for a final region merging step.

If $region_j$ and run_i do *not* have similar color values, the algorithm takes advantage of the run-length encoding: all other pixels in the previous row belonging to the same run are skipped (Figures 2(c)). By following this strategy, redundant comparisons are minimized.

3.1.3 Region Merging

The region merging step follows directly after a run run_i was added to a top adjacent region $region_j$. Any yet unconsidered regions are merged with $region_j$ if their color values are similar. When merging $region_j$ with an adjacent region $region_k$, $region_k$'s set of runs is added to $region_j$'s set of runs while the color values of $region_j$ are updated accordingly (Figure 2(g)). Again, the algorithm takes advantage of the run-length encoding: regardless of whether $region_j$ could be merged with a top adjacent region $region_k$ or not, redundant comparisons can be minimized by skipping all pixels belonging to the respective run (Figure 2(f)).

3.2 Robust Region Representation

Our segmentation algorithm produces a set of disjoint regions. Each of them, in turn, consists of a set of runs. For subsequent object detection steps, however, the pixel precise image data representation in terms of runs is inappropriate. In addition, pixel precise representation is prone to stochastic errors such as image noise. For that reason, we interpret a region and its associated pixels as two-dimensional Gaussian distribution in the image plane. Based on (Hu, 1962), the distribution is described by two moments m_{10} and m_{01} of first order (corresponding to mean values m_x and m_y) and three centralized moments μ_{20}, μ_{02}, and μ_{11} of second order (corresponding to variances σ_x^2 and σ_y^2 and covariance σ_{xy}). We already showed in (Jungmann et al., 2012) how to efficiently compute these moments by directly using the intermediate run-based representation and how to derive additional attributes such as center of mass or an equivalent elliptical disk.

Although assuming that pixels of a region are Gaussian distributed, the true distribution may be of arbitrary shape (like, e.g., the red region in Figure 2(h)). One approach for better approximating the true distribution might be the incorporation of moments of higher order. In our work, however, we currently consider only moments up to second order.

4 EVALUATION FRAMEWORK

For evaluating the robustness of our segmentation approach in combination with different color spaces, we set up the evaluation framework depicted in Figure 3. The entire framework is divided into two main sections: the actual image segmentation process and the evaluation process of the segmentation result.

The main component within the image segmentation section is our segmentation algorithm (cf. Section 3). It consumes a single RGB image and produces a set of regions, each of them described in terms of moments. The color space, in which the segmentation process should take place, is selected in advance. For each selectable color space, a heuristic for determining whether color values are similar, as well as a calculation specification for averaging color values are predefined. Each heuristic can be individually parametrized by means of thresholds. The necessary color space transformation from the original RGB color space into the selected color space is done on-the-fly during the segmentation process by means of an additional calculation specification.

The subsequent evaluation process can be divided into two branches. Both of them use the regions produced by the segmentation process to evaluate the robustness of the current segmentation setting. The first branch incorporates a tracking mechanism for evaluating relative robustness across consecutive images. The second branch uses ground truth data to evaluate absolute robustness for each frame individually.

4.1 Relative Robustness

Evaluating relative robustness of our segmentation algorithm in combination with a selected color space corresponds to evaluating how stable regions are detected among consecutive frames without moving the camera and without changes within the environment (except of illumination changes). By doing so, we want to estimate how good the different color spaces can deal with image noise as well as different illumination conditions.

4.1.1 Region Tracking

We apply a deterministic tracking approach that establishes correspondences between regions of consecutive frames. The amount of established correspondences is subsequently compared with the amount of regions that were originally extracted by the segmentation algorithm. The tracking algorithm was originally introduced in (Jungmann et al., 2010).

The main idea of the approach is to gradually reduce the amount of valid correspondences between a region extracted from image I_t and all regions extracted from image I_{t-i} by applying heuristics with respect to position, motion, size, and shape.

4.1.2 Evaluation

Let R_S denote the set of regions extracted by the segmentation algorithm. Furthermore, let R_T denote the set of successfully tracked regions detected by the tracking algorithm. We define the relative robustness κ_i for a framework iteration i as

$$\kappa_i = \frac{|R_{T_i}|}{|R_{S_i}|}. \tag{4}$$

In order to get a meaningful value for the relative robustness, we consider a sliding window comprising the last n framework iterations, and take the arithmetic mean

$$\kappa = \frac{1}{n}\sum_{i=1}^{n}\kappa_i = \frac{1}{n}\sum_{i=1}^{n}\frac{|R_{T_i}|}{|R_{S_i}|}. \tag{5}$$

If $\kappa = 1$, the current setting can be considered as highly robust, since corresponding regions were detected in each of the last $n-1$ images. If $\kappa = 0$,

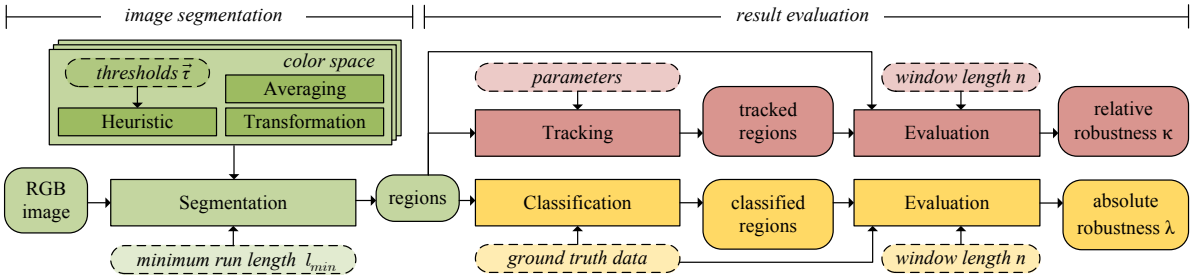


Figure 3: Our evaluation framework. It consists of a segmentation process and two evaluation branches for measuring relative and absolute robustness of the applied color space in combination with our segmentation algorithm.

the segmentation results significantly differ in consecutive images and the current setting cannot be assumed to be robust at all. Although R_T may contain regions with incorrect correspondences that distort the value of κ, we neglect an additional validation step, but minimize the probability of wrongly established correspondences by applying a highly restrictive parametrization to the tracking algorithm.

The final value of κ heavily depends on the parametrization of the selected heuristic. For example, by simple relaxing the heuristics's thresholds as much as possible, the entire image plane will be represented as a single region. Although the segmentation result is not useful for any further object detection steps, the relative robustness will be maximal with $\kappa = 1$. In short, κ alone is not sufficient for discussing the overall robustness of a segmentation setting. For that reason, we apply the additional evaluation branch based on ground truth data.

4.2 Absolute Robustness

Evaluating relative robustness is done without any foreknowledge. Evaluating absolute robustness, in turn, is based on manually generated ground truth data. By comparing segmentation results with ground truth data, we estimate how robust a color space can reproduce a desired result, e.g., under changing illumination conditions. In our work, ground truth data corresponds to a set of regions R_{gt} representing a given structure within an image. Providing that neither camera nor objects within the image area are moved, ground truth data has not to be generated for each image individually but once for an entire sequence. Pixels belonging to the same region are labeled and subsequently used to calculate the region representation in terms of moments (cf. Section 3.2).

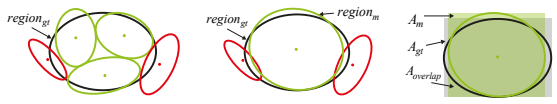


Figure 4: Classification and evaluation: (a) regions that lie within the boundaries of $region_{gt}$ are (b) merged into one single region $region_m$ in order to (c) determine the overlap between $region_{gt}$ and its related regions by means of their bounding boxes.

4.2.1 Region Classification

The evaluation process is based on a non-exhaustive classification step: an extracted region $region_j$ is assigned to a ground truth region $region_{gt} \in R_{gt}$, if its center of mass lies within the boundaries of $region_{gt}$ (cf. Figure 4(a)). More than one region may be assigned to $region_{gt}$, while $region_j$ may also be assigned to more than one ground truth region. Regions that are assigned to the same ground truth region are merged (cf. Figure 4(b)) by adding together the corresponding moments. As result of the classification process, each ground truth region $region_{gt} \in R_{gt}$ belongs to one (possibly merged) region $region_m$ at the maximum.

4.2.2 Evaluation

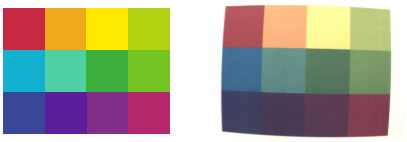
The main idea of the evaluation step is to analyze the overlap of a ground truth region $region_{gt} \in R_{gt}$ and its associated $region_m$. Instead of calculating a pixel-precise value, we consider the overlap of the region's bounding boxes based on their equivalent elliptical disks (cf. Figure 4(c)) and define the distance $\delta_{gt,m}$ between those two regions as

$$\delta_{gt,m} = 1 - \frac{2 \cdot A_{overlap}}{A_{gt} + A_m}, \tag{6}$$

with A_m being the area of $region_m$'s bounding box, A_{gt} being the area of $region_{gt}$'s bounding box and $A_{overlap}$ being the overlapping area of both bounding boxes. If $region_{gt}$ does not belong to any $region_m$, the distance has maximum possible value of 1. The absolute robustness λ_i for a framework iteration i is then defined as

$$\lambda_i = 1 - \frac{1}{|R_{gt}|} \sum_{k=1}^{|R_{gt}|} \delta_{gt_k,m}, \tag{7}$$

with $|R_{gt}|$ denoting the total amount of ground truth regions and $\delta_{gt_k,m}$ being the distance (6) of the k-th



(a) (b)

Figure 5: Color palette of 12 different colors: (a) original, synthetic version, (b) printed and captured by camera.

ground truth region and its assigned region. As output value λ, we consider the averaged absolute robustness of the last n framework iterations:

$$\lambda = \frac{1}{n}\sum_{i=1}^{n}\lambda_i \ . \tag{8}$$

5 EXPERIMENTS

We evaluated the robustness of the four different color spaces mentioned in Section 2 in combination with our segmentation algorithm (Section 3.1) by considering 14 image sequences each consisting of 200 images. All sequences show exactly the same scene but were acquired under different lighting conditions and camera settings, respectively. The scene itself consists of a palette of 12 different colors (Figure 5).

5.1 Color Space Specific Heuristics

We incorporated two different heuristics for deciding, whether two color values are similar or not. Heuristic h_1 is used for RGB and L*a*b*. It determines similarity of two color vectors $\vec{\mu}$ and $\vec{\nu}$ by checking the distance of each color channel individually:

$$h_1(\vec{\mu},\vec{\nu},\vec{\tau}) = \begin{cases} 1 & if \quad \begin{array}{l} |\mu_1-\nu_1| \le \tau_1 \wedge \\ |\mu_2-\nu_2| \le \tau_2 \wedge \\ |\mu_3-\nu_3| \le \tau_3 \end{array} \\ 0 & else \ , \end{cases} \tag{9}$$

with $\vec{\tau}$ being distinct threshold values. Heuristic h_2 is used for HSV and LCCS. Contrary to h_1, it checks the distance of only two color channels individually:

$$h_2(\vec{\upsilon},\vec{\omega},\vec{\tau}) = \begin{cases} 1 & if \quad \begin{array}{l} |\upsilon_1-\omega_1-\eta| \le \tau_1 \\ \wedge\, |\upsilon_2-\omega_2| \le \tau_2 \end{array} \\ 0 & else \ , \end{cases} \tag{10}$$

with

$$\eta = \begin{cases} 360 & if \ |\upsilon_1-\omega_1| \ge 180 \\ 0 & else \end{cases} \tag{11}$$

being a correction value. Again, $\vec{\tau}$ denotes a vector of distinct threshold values. In case of HSV color space, the V channel is neglected. In case of LCCS, the Cartesian coordinates (Equation 1) are transformed into polar coordinates.

5.2 Framework Settings

The applied values for the different control values are depicted in Table 1. The color space specific values for $\vec{\tau}$ were determined by feeding an image sequence with manually optimized camera parameters and constant lighting conditions to the framework. The threshold values were manually adjusted during execution until the maximal possible sum $\kappa+\lambda$ of relative and absolute robustness was identified.

Table 1: Static framework settings.

minimum run length	$l_{min} = 5$
window length	$n = 200$
RGB thresholds (h_1)	$\vec{\tau}_{RGB} = (13,14,8)$
L*a*b* thresholds (h_1)	$\vec{\tau}_{L^*a^*b^*} = (15,4,6)$
HSV thresholds (h_2)	$\vec{\tau}_{HSV} = (13,32)$
LCCS thresholds (h_2)	$\vec{\tau}_{LCCS} = (16,8)$

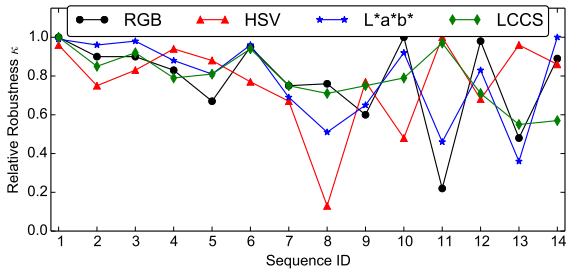
5.3 Results

In Figure 6, relative and absolute robustness of the 14 different illuminated image scenes are plotted (cf. short description in Table 2). Figure 6(a) shows that relative robustness strongly differs not only between different sequences, but also for segmentation with different color spaces applied to a single sequence. For sequences 1-7, where camera parameters are manually optimized for human perception, κ is in a high value range and differs maximal around 0.2 per sequence. This indicates that our segmentation algorithm provides mostly equal regions over time even at varying illumination (sequences 6 and 7). At sequences 8-14 camera parameters are manipulated, e.g., to get red, blue, or green cast. κ indicates that these manipulations have heavy impact on segmentation results depending on the applied color space. At blue cast in images, e.g., RGB and L*a*b* get problems while segmentation results with HSV and LCCS keep constant over time.

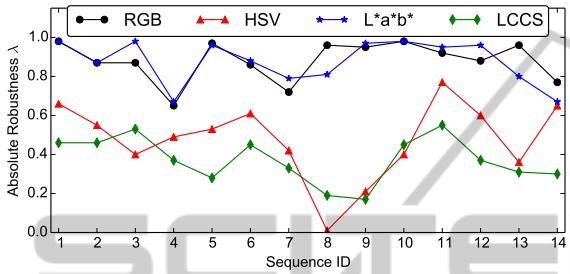
But relative robustness κ indicates only robustness of segmented regions over time. It does not consider whether segmentation results correspond to original

Table 2: Description of sequences considered at evaluation.

1	Manual optimized camera parameters	8	Room light off; spotlight from behind
2	Reduced brightness (flickering appears)	9	Room light off; spotlight from front
3	Reduced brightness	10	Red cast
4	Shadow	11	Blue cast
5	Spotlight	12	Green cast
6	Moving shadows	13	Minimum brightness of camera
7	Moving spotlight	14	Maximum brightness of camera



(a) Relative robustness κ.



(b) Absolute robustness λ.

Figure 6: Relative and absolute robustness results for all sequences described in Table 2.

color segments, i.e. ground truth data. Hence, we also have to consider absolute robustness λ depicted in Figure 6(b). For our segmentation algorithm, it indicates that robustness regarding segmentation of all present colors under varying illumination is higher using RGB or L*a*b* than applying HSV or LCCS. This is also confirmed by the examples shown in Figure 7. For the image scene with manual optimized camera parameters (Figures 7(a)-7(e)), we notice that usage of RGB or L*a*b* provides separate regions for each color of the palette. In contrast, segmentation with HSV combines, e.g., the red and orange field as well as the yellow and light green field. This might be caused by the non-linear color-ordering within the hue-axis where yellow is only represented by a small angle range while red and green have a higher resolution in HSV (cf. Figure 1(b)). Using LCCS provides an even less sophisticated segmentation result where, e.g., orange, yellow, and light green are combined.

Figures 7(f)-7(o) show that changes in illumination affect the segmentation results of all considered color spaces. Again, RGB provides best results and L*a*b* also performs well. Even though L*a*b* regions contain less segmented pixels, regions are similar to those of RGB. Within a vision system, results of segmentation would be used for post-processing steps, i.e. the representation of regions would be considered. Hence, results of segmentation with RGB and L*a*b* provide similar quality in the context of a vision system. In contrast, results of HSV and LCCS can, e.g., no longer cover the dark colors in presence of shadow (white areas in Figures 7(h) and 7(j)). In case of segmentation with HSV, this might be an implementation issue because within HSV we do not consider values positioned in a certain range around the vertical axis of the color cylinder to avoid noised gray values. Consequently, colors that appear almost gray or even black were neglected by segmentation using HSV. This has also negative effects regarding the absolute robustness because the distance function introduced in Equation 6 returns $\delta = 0$ for these missing color fields.

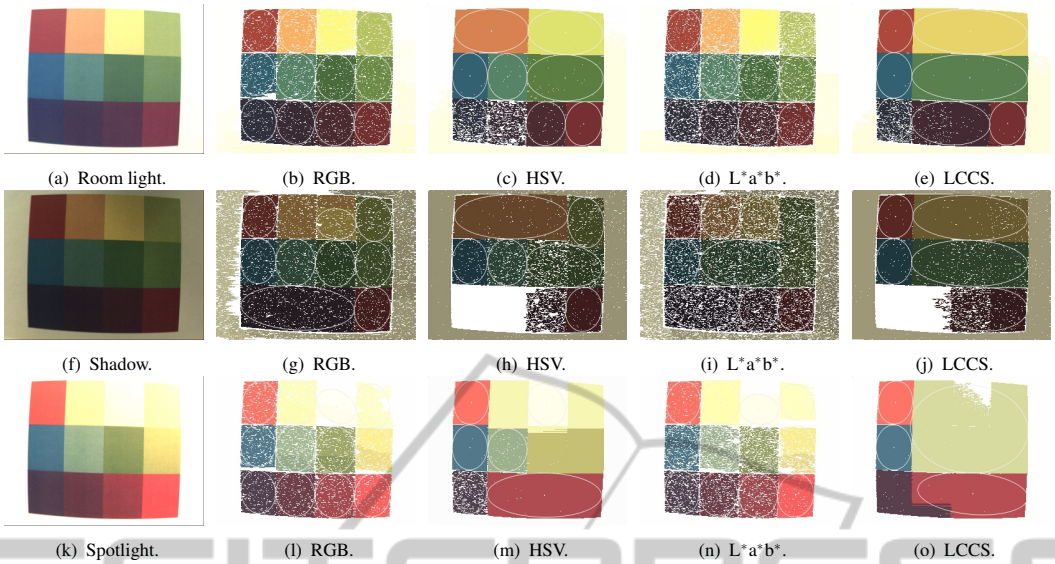
In presence of an additional spotlight (Figures 7(k)-7(o)), the yellow field of the palette is too much affected by reflections and therefore appears almost white in the original image. Nevertheless, segmentation with all considered color spaces except LCCS extracts a region that at least partially covers this area. Within segmentation using LCCS, all originally green and yellow color fields of the palette are fused to one region with kind of mint green as average color. Again, RGB and L*a*b* provide best results.

6 CONCLUSIONS AND OUTLOOK

In this paper, we presented our color-based segmentation approach and evaluated its robustness in combination with different color spaces. To investigate the robustness of a specific segmentation setting, we introduced our framework that directly works on our statistical region representation. In this context, the out-standing feature of the framework is the evaluation of two different types of robustness: relative robustness and absolute robustness. Relative robustness is evaluated by tracking regions across consecutive images, while absolute robustness is evaluated for each frame individually based on manually generated ground truth data.

With the evaluation of our segmentation algorithm, on the one hand we proved the relative and absolute robustness as a significant quantitative measure for robustness. On the other hand, we showed that in our artificial setting the presented segmentation algorithm performs best using RGB color space. Nevertheless, for some situations the segmentation approach provided better results using other color spaces like L*a*b* or HSV.

In our future work, we want to use the evaluation framework for automatically calibrating our segmentation algorithm as well as adapting the threshold values used by our heuristics during execution time. Furthermore, we want to switch from artificial scenes to more realistic scenarios while appropriately adjusting



(a) Room light. (b) RGB. (c) HSV. (d) L*a*b*. (e) LCCS.

(f) Shadow. (g) RGB. (h) HSV. (i) L*a*b*. (j) LCCS.

(k) Spotlight. (l) RGB. (m) HSV. (n) L*a*b*. (o) LCCS.

Figure 7: Original images and corresponding segmentation results using different color spaces. Images captured under room light (a-e; sequence 1), with additional shadow (f-j; sequence 4), and with a spotlight (k-o; sequence 10).

the functionality of our evaluation framework, if necessary. Aside from evaluating robustness of segmentation results, we additionally intend to investigate if the presented approach can be used for comparing images not on pixel level but on region level.

ACKNOWLEDGEMENTS

This work was partially supported by the German Research Foundation (DFG) within the Collaborative Research Center "On-The-Fly Computing" (SFB 901).

REFERENCES

Blas, M., Agrawal, M., Sundaresan, A., and Konolige, K. (2008). Fast color/texture segmentation for outdoor robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4078–4085.

Busin, L., Shi, J., Vandenbroucke, N., and Macaire, L. (2009). Color space selection for color image segmentation by spectral clustering. In *IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 262–267.

Finlayson, G. D., Drew, M. S., and Lu, C. (2009). Entropy minimization for shadow removal. *Int. J. Comput. Vision*, 85(1):35–57.

Freixenet, J., Muñoz, X., Raba, D., Martí, J., and Cufí, X. (2002). Yet another survey on image segmentation: Region and boundary information integration. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 408–422.

Hu, M.-K. (1962). Visual pattern recognition by moment invariants. *IEEE Transactions on Information Theory*, 8(2):179–187.

Jungmann, A., Kleinjohann, B., Kleinjohann, L., and Bieshaar, M. (2012). Efficient color-based image segmentation and feature classification for image processing in embedded systems. In *Proceedings of the 4th International Conference on Resource Intensive Applications and Services (INTENSIVE)*, pages 22–29.

Jungmann, A., Stern, C., Kleinjohann, L., and Kleinjohann, B. (2010). Increasing motion information by using universal tracking of 2d-features. In *Proceedings of the 8th IEEE International Conference on Industrial Informatics (INDIN)*, pages 511–516.

Khanal, B., Ali, S., and Sidib, D. (2012). Robust road signs segmentation in color images. In *Proceedings of the 7th International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 307–310.

Mogelmose, A., Trivedi, M., and Moeslund, T. (2012). Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497.

Russell, S. and Norvig, P. (2010). *Artificial Intelligence - A Modern Approach*. Pearson, 3 edition.

Szeliski, R. (2011). *Computer Vision - Algorithms a. Applications*. Springer.

Unnikrishnan, R., Pantofaru, C., and Hebert, M. (2007). Toward objective evaluation of image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):929–944.

Vandenbroucke, N., Macaire, L., and Postaire, J.-G. (2003). Color image segmentation by pixel classification in an adapted hybrid color space: application to soccer image analysis. *Computer Vision and Image Understanding*, 90(2):190–216.

Yang, U., Kim, B., and Sohn, K. (2009). Illumination invariant skin color segmentation. In *4th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 636–641.