Freie Universität Berlin
Fachbereich Mathematik und Informatik

Master's Thesis

Efficient Superpixel Creation in High-resolution Images by Applying a PLANT

Jan Draegert

Advisors:

Prof. Dr. Daniel Göhring
Prof. Dr. Raúl Rojas

July 14, 2017

Declaration of Academic Honesty

I hereby declare that this master's thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute and has not been published. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given in the bibliography.

Berlin, July 14, 2017

Jan Draegert

Abstract

Autonomous robots attempt to understand their environment in real-time. The data structure PLANT, which this thesis introduces, clusters large amounts of data efficiently. To detect homogeneous image regions fast, a PLANT combines an integral image with a binary search. Moreover, a PLANT-based vision for soccer-playing robots is presented. It enables the robots to perceive their environment with a camera resolution of 1920×1080 pixels at a frame rate of 30 Hz. For superpixel creation, the algorithms `PLANT` and `PLANTm` are introduced. The time complexities of `PLANT` and `PLANTm` are $\mathcal{O}(n+k\cdot\log(n))$ and $\mathcal{O}(n+k\cdot\log(n)+k^2)$, where n is the number of pixels and k the number of superpixels. PLANT-based algorithms benefit from the spatial locality of reference, which results in a high speed-up.

`PLANT` and `PLANTm` are compared to state-of-the-art superpixel algorithms. In the experiments, no other state-of-the-art algorithm but `PLANT` and `PLANTm` achieved a frame rate of 30 Hz at a HD resolution. At a resolution of 3504×2336 pixels, `PLANT` required 21 ms, while the fastest non-PLANT-based algorithms, whose parameter were optimized for efficiency, required 368 ms. The achieved superpixel quality is comparable with state-of-the-art algorithms—and similar to the watershed segmentation algorithm. Thus, the use of `PLANT` and `PLANTm` is recommended if superpixels have to be created on high-resolution images fast.

Contents

1 Introduction

Contemporary robots are certainly not particularly intelligent. Nonetheless, as long as a robot performs its designated task and no unforeseeable event occurs, that robots can appear to have some brainpower, provided that it acts and reacts properly. To react properly, the robot has to perceive information from its environment and draw apposite conclusions. Moreover, the robot can use perceived information to learn and thus make better decisions in the future.

Electromagnetic waves are able to carry information. Interestingly, almost all knowledge of outer space originates from the interpretation of quantity, direction, and frequency of electromagnetic waves, which are observed over time. Furthermore, a high proportion of the human brain is specialized for decoding information from visual signals [91]. Most autonomous robots, especially those that operate in an environment that is made for humans, are equipped with cameras as a part of their sensor system—because essential information can be extracted from light waves. While more sensitive telescopes are still built [33], and the human visual cortex continues evolving [72], affordable high-resolution cameras have become obtainable. Yet, for the sake of real-time processing, many robots do not employ high camera resolutions.

Real-time capability is crucial for most autonomous robots since they interact with the real world. Image segmentation is an established method in computer vision to abstract data and thus speed up subsequent processing steps. Given a segmentation, it can be decided which segments need further processing and what kind of processing these segments need. However, a drawback of image segmentation is that object boundaries can get lost with the result that one segment contains pixels from more than one object. In contrast, superpixels adhere to boundaries and group pixels into natural entities so that essential information about the structure in the image are preserved [76]. An object may consist of several superpixels, but no superpixel should contain pixels of more than one object. Superpixels do not only speed up subsequent processing steps; to be beneficial, they themselves are computed fast. There are many superpixel algorithms. Unfortunately, there has been no algorithm so far that creates superpixels in high-resolution images and runs on a customary computer in real-time.

It is difficult to achieve real-time on high resolutions because any algorithm that considers each pixel runs at least in asymptotic linear time as a function of the number of pixels. This thesis employs integral images [98] to create superpixels highly efficiently by using the hierarchy of memory. The integral images are computed in linear time, yet in a fast manner. All subsequent parts of the algorithm, which may access the memory more randomly, run in asymptotic logarithmic or constant time. More precisely, the major contributions of this thesis are summarized as follows:

- The data structure PLANT, which is designed to partition data in real-time, is introduced (see chapter 3).
- It is explained how a PLANT can be used to create superpixels (see chapter 3).
- It is outlined how a PLANT is employed to implement a vision for soccer-playing robots (see chapter 4).
- The segmentation quality of the PLANT-based superpixel creation method is compared to state-of-the-art superpixel algorithms (see chapter 5).
- The run times of the PLANT-based method and of fast state-of-the-art algorithms are compared on several, especially high, image resolutions (see chapter 5).

2 Preliminaries

Similar to most cameras, the cameras of the soccer-playing robots, for which a computer vision is implemented in chapter 4, provide a stream of JPEG images that represent colors using the YCbCr model. Section 2.1 explains how and why the YCbCr model is useful. To process the YCbCr data, a PLANT, which is introduced in chapter 3, is generated. Integral images, covered in section 2.2, are the key technique for generating a PLANT efficiently. Section 2.3 introduces superpixels. In chapter 5, the efficiency and quality of superpixels generated by a PLANT are compared to the superpixels algorithms presented in section 2.4.

2.1 Color Representation

2.1.1 RGB Color Model

The RGB color model is based on the Young-Helmholtz theory [100], which have been proved true for the most parts. The theory states that light consisting of three different wave lengths, which are the primary colors, can form all arbitrary colors within the wave lengths' spectrum. The resulting color depends on the intensities of the primary colors. Von Helmholtz presumed that the human eye possesses three types of receptors to perceive all colors. Shades of gray are perceived when the light excites the three receptor types with the same intensity. If none of the receptors is stimulated, black is "perceived".

Von Helmholtz assumed that the three types of receptor cells are most sensitive to red, green, and blue (nowadays, it is known that the peaks are at slightly different wavelength). In the RGB color model, a color is therefrom specified by the intensities of the primary colors red (R), green (G), and blue (B). Each of the three color channels is usually encoded by one byte. Therefore, a RGB triplet represents $(2^8)^3 = 16\,777\,216$ different colors. The RGB color model is commonly applied in display devices. Usually, displays consist of three types of light-emitting diodes or liquid crystals that emit red, green, or blue light. There are several color spaces, including sRGB and Adobe RGB, that specify the three primary colors. The primary colors in turn define the color gamut, i.e. the individual colors that the RGB triplet represents.

2.1.2 YCbCr Color Model

The brightness perception of humans is superior to their perception of color. The visual system detects finer details of brightness variations as it consists of more rods than cones. Accordingly, JPEG subsamples color values for compression. Since brightness and color values correlate in the individual RGB channels, JPEG uses the YCbCr color model, where brightness and color are separated. There are several notations and definitions for the YCbCr color model; this thesis complies with the JPEG File Interchange Format [38]. The YCbCr color model encodes brightness into a luminance channel (Y) and color into two chrominance channels (Cb, Cr).

Initially, a gamma correction is applied to the RGB values [44]. The gamma correction compensates a bias of the human perception: According to the Weber-Fechner law, humans are able to sense smaller differences of a physical stimulus if the stimulus is less intense [28]. As a result of the gamma correction, distances of colors become more similar to the human perception. Afterwards, the three 8-bit YCbCr values are computed as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix} \tag{2.1}$$

Equation 2.1 illustrates that the color green contributes most to the luminance value. The chrominance values are the differences between the luminance and the channels that are less correlated to brightness, i.e., B − Y yields Cb and R − Y yields Cr. The conversion from RGB to YCbCr enables JPEG to subsample the chrominance components. There are multiple subsampling modes. For example, 4:2:0 subsampling omits every second chrominance pixels in both horizontal and vertical direction; consequently, the resolution of both chrominance channel is quartered, and thus the size of the entire image is halved.

2.2 Integral Images

Integral images originate from texture mapping [20] in computer graphics, where they are referred to as “summed area tables”. Viola and Jones [98] have applied them for object detection and termed them “integral images” for the usage in the field of computer vision. While integral images are usually used to compute rectangular features for supervised object detection, this thesis employs them for unsupervised detection of general object boundaries.

2.2.1 Pre-computation

To compute the integral image I_Σ, all pixels that are located above and to the left of the corresponding pixel in the input image I are summed up:

$$I_\Sigma(x,y) = \sum_{i \le x} \sum_{j \le y} I(i,j) \tag{2.2}$$

In practise, the integral image is computed via a single pass over the image. For this purpose, both an additional row and an additional column, which are filled with zeros, are put in front of the first row and the first column in the integral image. Moreover, a pointer, pointing to the current column in the previous row, and an accumulator S, summing up the current row's values, are defined. The accumulator is initialized to zero for each row. While iterating over a row, the accumulator sums pixel values up (see Equation 2.3). The value of a pixel in the integral image is obtained by adding the accumulator and the dereferenced pointer, which yields the pixel value of the integral image in the row above and in the same column (see Equation 2.4).

$$S(x,y) = S(x-1,y) + I(x,y) \tag{2.3}$$

$$I_\Sigma(x,y) = I_\Sigma(x,y-1) + S(x,y) \tag{2.4}$$

2.2.2 Application on Images

The integral image is used to compute the sum or mean of any rectangular image region with a time complexity of $\mathcal{O}(1)$. Each pixel in the integral image represents a rectangular region. To calculate the pixel sum of a region, initially, the right bottom corner of the region is considered. The integral value of this corner represents an area that covers the complete region. Nevertheless, this area potentially contains pixels located above and to the left of the region. These pixels are removed by subtracting the integral values of the bottom left corner and the upper right corner. Consequently, the area represented by the upper left has been subtracted twice; thus, it is added once again. In summary: Suppose the upper left, upper right, bottom left, and bottom right corners of the region are denoted by $A = (x_0, y_0)$, $B = (x_1, y_0)$, $C = (x_0, y_1)$, $D = (x_1, y_1)$, respectively, the region's pixel sum $\sum_{\substack{x_0 \le x \le x_1 \\ y_0 \le y \le y_1}} I(x,y)$ is calculated as follows:

$$\sum_{\substack{x_0 \le x \le x_1 \\ y_0 \le y \le y_1}} I(x,y) = I_\Sigma(D) - I_\Sigma(C) - I_\Sigma(B) + I_\Sigma(A) \tag{2.5}$$

Equation 2.5 illustrates that three additions or subtractions and four references to the integral image are required to compute any region's sum. Eventually, the sum is divided by the region's area to obtain the region's mean value.

2.3 Superpixels

2.3.1 Why Superpixels?

A segmentation partitions an image into its integral components to facilitate further processing. However, segmentation is an ill-posed problem because it has no unique solution; Figure 2.1 illustrates this. Superpixels can be regarded as a result of an unsupervised over-segmentation [66]; they avoid under-segmentation to preserve most segment boundaries of various ground truth data. Simultaneously, superpixel algorithms preferably create a small number of superpixels to capture redundancy.

Before superpixels were established, Shi and Malik [84] had presented a Bayesian view on segmentation: The interpretation of the quality of an segmentation is based on prior knowledge. Prior knowledge comprises not only low-level cues such as coherent brightness, color or texture, but also mid- and high-level cues. Higher-level cues combine segments computed by low-level cues. In contrast to low-level cues, they can consider context, instead of only relying on intrinsic object information; if the context is neglected, it can result in a bad segmentation on natural images due to occlusion, bad illumination, and shadows [90]. Therefore, a good segmentation is inherently hierarchical. This insight gave rise to superpixels, which are computed through low-level cues and enable fast higher-level processing [76].

Accordingly, superpixel creation is a preprocessing step, reducing the complexity of an image without loosing much information. Superpixels speed up subsequent computations significantly. For example, a graph-based segmentation algorithm runs notably faster if the graph's nodes consist of several hundred superpixels instead of several hundred thousand pixels. Pixels contain redundant data because they are a result of a discrete representation of an image, while superpixels are more natural entities, resulting from perceptual grouping [68].

2.3.2 Range of Applications

Superpixels have been used in a wide range of image processing and computer vision tasks. Most of the work about superpixel applications has been published in recent years. Originally, superpixels are intended to improve image segmentation in general [76]; there have been several enhancements [40,49,65,110]. Superpixels are also used for interactive segmentation, where a user adapts segmentation boundaries in real-time [47,83]. In some cases, each pixel has to be classified, i.e. each pixel obtains an object label (semantic segmentation [32,36,51,89,108]) or each pixel is assigned to either a specific object or the background (object segmentation [7,16,22,34,59,75,102]). In other cases such as object recognition [92] or object detection [57, 85, 109], including human detection [67] and path detection [21], it is sufficient to approximate the object's position with a bounding box.

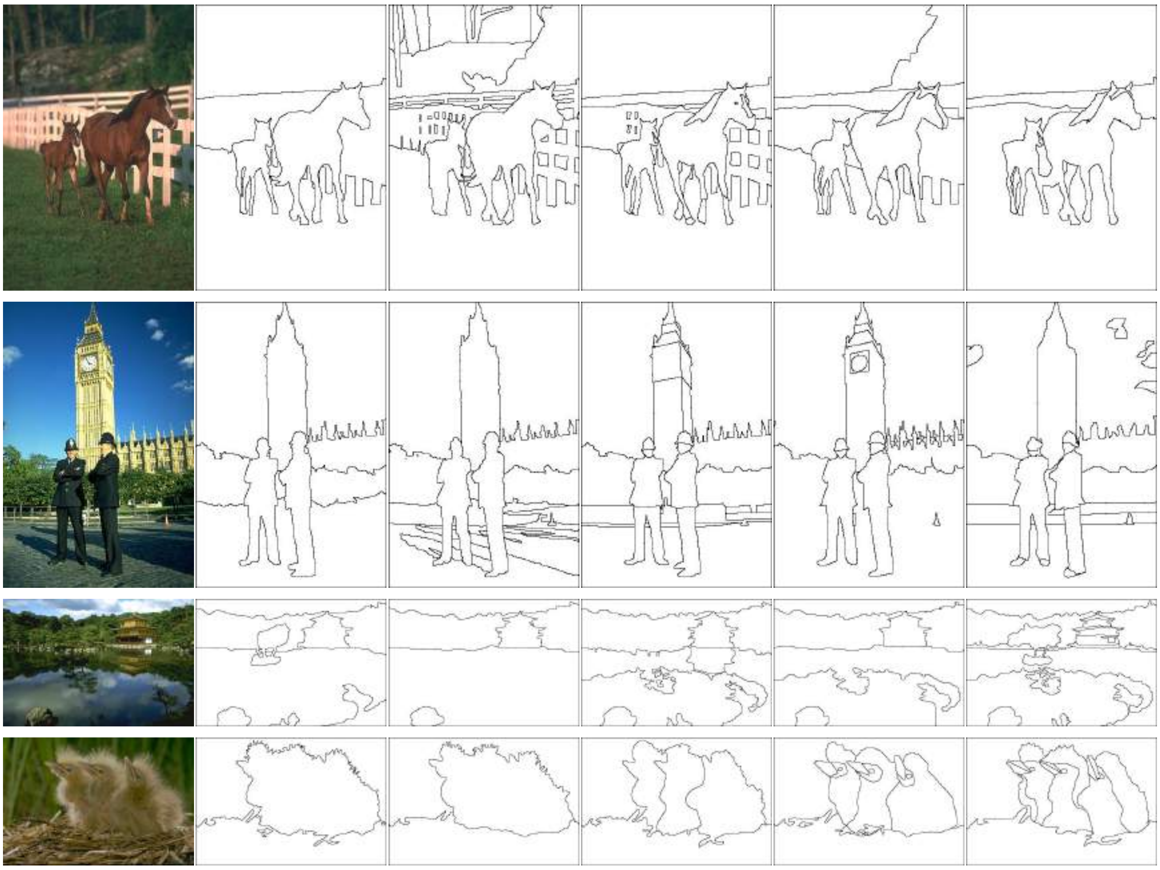


Figure 2.1: Segmentation is an ill-posed problem: When humans are asked to segment images, their solutions appear reasonable but different from each other. It is difficult to induce a segmentation that is adequate for any purpose. Superpixels avoid this difficulty through over-segmentation. The images in the leftmost position are the original images. The images to the right are ground truth data, which were drawn by different humans. All images are part of the Berkley Segmentation Dataset 300 [61].

Saliency detection is a prominent application for superpixel algorithms [3, 39, 70, 73, 103, 113, 114]. These algorithms detect whether a pixel belongs to a salient image region that captures the viewers' attention. Furthermore, superpixels also help to build a 3D-model out of a single image; they facilitate 3D-reconstruction [12, 41, 80] and depth estimation [52, 54], which, for instance, is used for indoor scene understanding [37, 50]. Superpixels are similarly employed for optical flow and stereo matching [58,107]; by combining both, scene flows [63,99] can be computed as well. Other fields of application are data set annotation [56,108], tracking [8,101,111] and the acceleration of convolutional neural networks [39, 52].

2.4 Related Work

2.4.1 Superpixel Algorithms

There is a considerable number of superpixel algorithms. The superpixel benchmark of Stutz et al. [88] has given a comprehensive overview of current state-of-the-art superpixel algorithms and has both evaluated and compared them. Beyond that, the superpixel benchmark has subdivided the algorithms into several categories. Graphed-based, watershed-based, clustering-based, energy optimization, contour evolving, density-based and path-based algorithms are relevant for this thesis. This classification is not strict; several algorithms exhibit features of multiple categories. Selected algorithms from which a low run time is to be expected are evaluated in chapter 5 and introduced in this section. In the interests of brevity and clarity, acronyms in typewriter font are assigned to these algorithms. The acronyms comply with the acronyms used by Stutz et al., they are used in the remaining thesis. For instance, the superpixel creation algorithm that is based on a PLANT is typeset as `PLANT`[1].

Ren and Malik [76] introduced the term "superpixel" in their paper about normalized graph cuts [84] that produce an over-segmentation. The classical Gestalt psychology [104] had inspired them with its description of perceptual grouping. Normalized graph cuts have enhanced the original graph cut algorithm [105], which has a bias towards splitting smaller segments more likely into even smaller segments while neglecting larger segments. Normalized cuts have fixed this flaw by attaching a normalization term. The original normalized cut algorithm was rather slow, but there have been several efficiency optimizations [19, 25, 106].

Felzenszwalb and Huttenlocher (`FH` [29]) have introduced a graph-based method, which is faster than normalized cut algorithms. It uses Kruskal's algorithm to build a minimum spanning tree. The graph's vertices represent pixels and the edges measure dissimilarities between vertices. The regions, which are vertices, are grown by merging similar regions

[1] Unlike the superpixel algorithm `PLANT`, the data structure PLANT itself is not typewritten (chapter 3 explains the difference).

greedily in a bottom-up manner. The entropy rate superpixels method (ERS [55]) has improved the quality of the bottom-up region merging used in FH significantly. ERS maximizes an objection function that is based on the entropy rate of a random walk on the graph. Unfortunately, this increases the run time. As opposed to this, Pseudo-boolean optimization (PB [115]) is a faster graph-based algorithm. It was inspired by Veksler et al. [95], who had assigned pixels to overlapping squares. Other than Veksler et al., PB puts vertical and horizontal strips on the image to obtain a superpixel lattice. Using the elimination algorithm [15] for optimization, PB processed several frames per second.

Watershed-based algorithms are fast—while providing a segmentation of good quality. First of all, the watershed algorithm [97] computes a gradient image and interprets it as a topological terrain. In a metaphorical way, the terrain is flooded successively. Consequently, the edges disappear underneath the water surface. The remaining gradients yield the segment boundaries. This thesis considers several variations of the watershed algorithm: Meyer has modified watershed for color images (W [64]); Neubert and Protzel have proposed compact watershed (CW [69]), which distributes seed points on the image to receive a compact segmentation; and Benesova and Kottman have presented a morphological superpixel segmentation (MSS [9]), which uses morphological reconstruction [96] to remove local extrema. The latter indicated to be efficient for higher resolutions.

Simple linear iterative clustering (SLIC [2]) has adapted k-means clustering for superpixel creation. The algorithm starts from a regular grid of initial points and lets superpixels grow around these points. After the growing step, the update step moves the points to the central position of their respective region. There are usually several iterations of the growing/update-cycle in order to refine the superpixels. SLIC is rather fast; however, preemptive SLIC (preSLIC [69]) speeds it up for real-time applications. It preempts iterations that miss a significant change of their points in the update step. Linear spectral clustering (LSC [48]) is another k-means-based algorithm. LSC applies clustering to normalized cuts and thus reduced the run time of normalized cuts algorithms from $\mathcal{O}(n^{\frac{3}{2}})$ to $\mathcal{O}(n)$, where n is the number of pixels.

Van den Bergh et al. have presented superpixels extracted via energy-driven sampling (SEEDS [93]). Just as SLIC, it starts from a regular grid of seed points. Since the growing step slows the superpixel segmentation down, SEEDS saves this step. Instead of this, SEEDS considers only the superpixel boundaries. It uses hill-climbing optimization to refine the boundaries. A revisited version of SEEDS (reSEEDS [87]) affixes a compactness term to the optimization process. Thus, reSEEDS achieved higher connectivity among superpixels and reduced run time. Extended topology preserving segmentation (ETPS [112]) is based on SEEDS. It stores pixels to be updated in a priority queue to realize a coarse-to-fine segmentation. ETPS has been ranked as the top-performing algorithm in the superpixel benchmark on all five studied data sets [88].

TurboPixels [46] is a representative for contour evolution algorithms. It computes geometrical flows to grow regions. Once again, seed points are placed regularly at the beginning. The contours of the segments are evolved iteratively. In each iteration the evolution speed is decreased; it converges to zero until no further evolution is possible.

TurboPixels has been sped up by eikonal region growing clustering (ERGC [14]). ERGC uses a fast merging method and was able to segment images obtained by CT scanning in real-time.

Comanicui and Meer [18] have proposed a density-based algorithm that applies mean shift for mode-seeking to a density image. Quick shift (QS [94]) decreased the run time of the mode-seeking step by using the faster Euclidean medoid shift [82] instead of mean shift. PathFinder (PF [24]) is a representative for path-based superpixel algorithms. It uses dynamic programming to compute minimum-cost paths along boundaries. The superpixel benchmark [88] reported that PF is one of the fastest superpixel creation algorithms available.

This thesis focuses on reducing computation time using a customary CPU. Nevertheless, it is worth mentioning that GPU implementations [11, 77] of superpixel algorithms enabled a high speed-up by more than an order of magnitude at high resolutions. However, PLANT enables similar run times without requiring a GPU architecture by employing integral images. Whereas there are a few approaches that use integral images for various steps in the image segmentation process [1, 6, 13, 30], current state-of-the-art superpixel algorithms do not use them.

2.4.2 Vision of Soccer-playing Robots

Successful robot soccer teams reach a frame rate of 30 Hz for all computation. As computer vision is computationally expensive, they have only considered pixels on scan lines or grid lines and thus have skipped data [62, 74, 78], or they have run the vision at a low resolution [4, 27, 62, 78]. As a result, the robots have not been able to recognize the ball across the field.

The FUmanoids already have used integral images for their vision [81]. For each channel of the YCbCr color model, they calculate an integral image. The colors are classified into eight logical colors [79] and a integral histogram, which consists of eight integral images, is calculated. They detect objects through a binary search, which generates a binary tree [23, 71]. Each node of the tree refers to an image region and the detected object consists of, possibly merged, leaves. With this approach, the usage of a resolution of 960×720 pixels at a frame rate of 30 Hz became achievable—while considering every pixel in the image. The PLANT-based vision that is presented in chapter 4 omits the computation of the integral histogram and only computes the integral image of the three YCbCr channels. As a consequence, the robots can process even higher image resolutions.

Holz et al. [42] have implemented a real-time plane segmentation of point clouds. They model objects through planes, where normals of the planes are estimated by smoothing them with image patches that are calculated out of integral images. The FUmanoids have studied the combination of plane segmentation and binary search for object detection in point clouds [86]. For this, they have built a tree of planes whose nodes split by estimating a proper split location and direction. Unfortunately, the tree generating procedure showed some indirections and frequently met impasses in several local minima. In contrast, a PLANT is more consistent and prevents this, which results in completely different segmentations. Accordingly, the way of building the partition tree is designed entirely different in chapter 3.

2.4.3 Summary

In short, there are numerous superpixel algorithms that strive for reducing run time while producing applicable superpixels. There has been no implementation of a superpixel algorithm on customary hardware that achieves real-time frame rates for processing high-resolution images. The FUmanoids have employed integral images that enable the processing of a large number of pixels. This thesis continues the idea of using integral images by suggesting a superpixel algorithm that can be applied to large images efficiently (see chapter 3).

3 Superpixel Creation Using a PLANT

This chapter defines the data structure PLANT and describes how this data structure is generated. Furthermore, it is explained how a PLANT is used to create superpixels. Two major superpixel creation techniques, `PLANT` and `PLANTm`, are presented. For both `PLANT` and `PLANTm`, this chapter discusses characteristic superpixel properties.

3.1 What is a PLANT?

The objective of building a partition-locating, axis-aligned, and node-queuing tree—a PLANT—is to obtain leaves that enclose homogeneous regions. Initially, a PLANT consists solely of one root node. The root node contains a box, which is a multidimensional interval, that encloses exactly the data to be analyzed. To grow a PLANT, nodes split into further nodes. A priority queue manages which node splits next; it contains all current leaf nodes of the tree and pops the most heterogeneous node if requested. When a node splits, an axis-aligned hyperplane partitions the data into two half-spaces. Each of the box-shaped half-spaces belongs to one of the two newly created nodes. The hyperplane is fitted into a selected location for the purpose of separating dissimilar data. The building process completes as soon as the heterogeneity of all nodes fall below a certain threshold or the PLANT consists of a predefined number of leaves.

A PLANT can be conceived as a binary space partitioning tree [31] that is similar to a k-d tree [10]. Notwithstanding, there are crucial differences between PLANTs and k-d trees. A k-d tree alternates the splitting direction successively and positions the hyperplane at the median data point in order to obtain two nodes featuring the same amount of data—regardless of the data's content. As opposed to this, a node of a PLANT splits in a certain direction and at a certain position so that the split creates two half-spaces containing data clusters that are as dissimilar as possible to each other. Moreover, the split maximizes the homogeneity of the resulting nodes' data. The metrics for dissimilarity, heterogeneity, and homogeneity are undetermined by design; they depend on the kind of data and the field of application.

3.2 Generating a PLANT

Without loss of generality, this section outlines the algorithm for two-dimensional images and describes the binary search for a split in horizontal direction; the algorithm is applicable for high-dimensional data and other split directions as well. For multi-channel images, only the two image dimensions are considered because partition planes that are orthogonal to the image plane are required. The individual channels can be regarded as individual features. A weighted arithmetic mean is used to combine the features; the weights are equal per default, but they are adjustable.

3.2.1 Partition Locating

The goal of the partition locating step is to locate the most significant edge in an image region for a determined direction. A binary search is used to find this edge quickly. Initially, the split locations is assumed to be at the center, so that a vertical line splits the image region into two halves of equal size. This temporary assumed split location is subsequently termed pivot location. A step size, which is initially set to a quarter of the image regions' width, is defined. Then, the pivot location is moved by the step size both to the left and to the right. Accordingly, one node consists of three quarters of the original image region, and the other node consists of the remaining quarter (see Figure 3.1). The difference of the average values of the respective resulting nodes is computed. If the difference that resulted from the movement to the left is higher than the difference that resulted from the movement to the right, the left pivot location remains, while the right pivot location is discarded. If it is lower, the right pivot location remains, while the left pivot location is discarded. In the unusual case that both differences are equal, the pivot location stays at its original center location.

For the next iteration, the step width is halved. Additionally, the half to which the pivot location has not moved to is cut off from the image region. Thereby, each iteration finds boundaries more locally than the previous iteration did. The pivot location is adapted iteratively as long as the step size is not less than four pixels. If it becomes less than or equal to three pixels, the averaging image gradient filter in horizontal direction $\bar{G}_x$ is applied for each of the remaining positions, and the pivot location is set to the position featuring the highest absolute filter response. The averaging gradient filter is defined as follows:

$$\bar{G}_x = \begin{bmatrix} -1 & -1 & \dots & -1 \\ 1 & 1 & \dots & 1 \end{bmatrix}^T \tag{3.1}$$

The number of rows in the non-transposed matrix $\bar{G}_x$ equals the height of the image region in pixels. Finally, the pivot location yields a vertical partitioning line that splits the heterogeneous image region into two more homogeneous regions.

(a) (b)

Figure 3.1: These both filters are used to calculate dissimilarities in each iteration. If the dissimilarity computed by the left filter is bigger than the dissimilarity computed by the right filter, the left pivot location is used in the next iteration. The same applies vice versa. Otherwise the pivot location does not change.

3.2.2 Node splitting

To identify the node that splits next, a PLANT employs a priority queue. The priority of a node is equal to its heterogeneity. The heterogeneity is already calculated while the above-mentioned partition location is searched. In each iteration of the search, an adapted pivot location is determined by the filters displayed in Figure 3.1. Each time the pivot location changes, its corresponding filter (Figure 3.1a, if pivot has moved to the left; Figure 3.1b, if pivot has moved to the right) yields a dissimilarity. This dissimilarity is weighted by a factor, which is halved in each iteration. The weighted dissimilarities are accumulated. At the end, the sum is multiplied by the node's length of the investigated direction (i.e. by the width for a search in horizontal direction). The result yields the directional heterogeneity. The heterogeneity of a node is the maximum of its directional heterogeneities. As soon as another node needs to be split, the priority queue pops the most heterogeneous node, and the node is split in its most heterogeneous direction at the precomputed split location.

3.3 Superpixel Creation

A straightforward option to create superpixels is to reinterpret each leaf of a PLANT as a superpixel. This approach will be referred to as `PLANT` in this thesis. An alternative approach, subsequently called `PLANTm`, is to merge the leaves of a PLANT into superpixels. Leaves that form a diagonal line are particularly often similar; therefore, they are expected to merge frequently.

To prepare the merging, the PLANT is traversed to find adjacent leaves for each leaf. The traversal starts at a initial leaf, successively visits all parent nodes up to visiting the root node, and visits all children that touch the initial leaf and have not been visit before. If a child is a leaf, it is neighbor. Once identified, the neighbors are attributed to their respective leaf; thus, the PLANT is not required to be traversed more than once per leaf. Adjacent leaves merge if their dissimilarity is below a certain threshold[1]. When leaves merge, their areas, their neighbors, and their properties merge as well. For example, the property "mean color" merges by averaging the colors and weighting the average by the area of their respective leaf. The merging is performed in several iterations. In each iteration, no leaf is involved in more than one merge with another leaf. This prevents unbalanced merging.

The intention of the merging procedure is to obtain a better segmentation for the same number of superpixels. In contrast, the disadvantage of merging is that it increases the run time for superpixel creation. A merged superpixel is compactly represented by a list of nodes[2] that contain boxes.

3.4 Properties of PLANT-Superpixels

3.4.1 Regularity

Several applications require regular superpixels such as applications using Markov random fields [26, 32, 36, 47]. Yet, there are multiple definitions of regularity. Giraud et al. [35] maintain that a regular superpixel features a smooth contour, a solid shape, and a balanced pixel distribution. Smoothness describes the ratio of the contour length to the length of its convex hull. Solidity specifies the ratio of the superpixels area to its convex hull's area. Balance considers the spatial distribution of pixels belonging to the superpixel: Within a well-balanced superpixel, the difference of the horizontal and vertical variance of pixel positions from the superpixels barycenter is small.

As `PLANT` produces rectangles, the superpixel's contour is identical to its convex hull. Hence, both smoothness and solidity are optimal. As mentioned above, weighted dissimilarities are accumulated to compute the directional heterogeneity. The weighted dissimilarities are multiplied by the length of the image patch in the respective direction. By weighting the length factor, balance can be adjusted. `PLANTm` produces superpixels whose convex hull is not necessarily identical to their contour. Furthermore, it disregards balance, although introducing a balance term appears possible if required. Eventually, `PLANTm` usually creates less regular superpixels than `PLANT`.

3.4.2 Connectivity

Connectivity means that each superpixel represents a connected set of pixels [46]. Some superpixel algorithms such as `SLIC` do not enforce connectivity [2]. The leaves of a

[1] If the dissimilarity threshold equals zero, `PLANTm` outputs the same superpixels as `PLANT`. Thus, `PLANTm` can be seen as a generalization of `PLANT`.
[2] For efficiency, the "list of nodes" is implemented as a `std::vector` of pointers to nodes.

PLANT are rectangles. Therefore, superpixels created by `PLANT` are inherently connected. In the merging step, only superpixels are joined that share a common boundary. That implies that at least one pixel of each original superpixel is connected with the respective other original superpixel; the resulting superpixel is thus connected as well. Consequently, both `PLANT` and `PLANTm` ensure connectivity.

3.4.3 Controllability

Controllability means that the number of superpixel are adjustable by a parameter [88]. Considering the superpixel algorithms that are compared to `PLANT` and `PLANTm` in this thesis, all but `FH` and `QS` provide such a parameter. Nevertheless, hardly any algorithms is accurately controllable. Almost all algorithms output a number of superpixels that is different from the preset number of superpixels. `ERS` is the solely algorithm of these that can be controlled accurately.

Thanks to its priority queue, which pops nodes one by one, `PLANT` is accurately controllable as well. In contrast, `PLANTm` is controllable, but not accurately controllable. Apparently, `PLANTm` could likewise employ a priority queue. Notwithstanding, it has to be studied how a priority queue can be applied reasonably since it is not clear how many nodes the other priority queue in the preceding splitting step should generate.

3.4.4 Efficiency

While the properties mentioned above are considered secondary in this thesis as PLANT has not been optimized for them, the principal goal of this thesis is to generate superpixels efficiently. More precisely, the focus is to enable a complete vision for soccer-playing robots (see chapter 4) to run on a stream of HD images with at least 30 frames per second. Superpixels facilitate this, but the superpixel creation cannot receive all run time available because it is a preprocessing step; subsequent computations require processing time as well. Moreover, superpixels are generated to accelerate subsequent computations. Therefore, efficiency is the essential property for superpixel algorithms.

`PLANT` and `PLANTm` are fast because they apply integral images. The time complexity of computing the integral image is $T(n) = c \cdot n = \mathcal{O}(n)$, where n is the number of pixels, and c is a constant factor. Compared to other superpixel algorithm, `PLANT` and `PLANTm` can be implemented so that c is notably small. Despite its name, random access memory (RAM) can be accessed faster sequentially than randomly because the memory hierarchy utilizes the spatial locality of reference. The constant factor c is small because memory cells are not accessed randomly but successively. The successive accesses particularly results in a high cache hit ratio as the prefetcher uses the regular access pattern to load suitable cache lines; furthermore, the cache lines are processed completely. Thus, run time does not only depend on the number of operations but also on the usage of the hierarchy of memory; `PLANT` and `PLANTm` consider both.

After each split, both resulting nodes are searched for a vertical and a horizontal partition location; it is a binary search, which has logarithmic time complexity. The binary search is applied for each node. A PLANT that generates k superpixels consists of $2 \cdot k - 1$ nodes including k leaves. Thus, the node creation has a time complexity of $\mathcal{O}(k \cdot \log(n))$ altogether. Consequently, the total time complexity for `PLANT` is $\mathcal{O}(n + k \cdot \log(n))$.

Although the merging procedure employs the tree structure of a PLANT, it is not ensured that the PLANT is balanced[3]. Thus, the time complexity for finding all respective neighbors is $\mathcal{O}(k^2)$ in the worst case. Afterwards, each leaf is compared to its neighbors; therefore, for the comparison part, $\mathcal{O}(k^2)$ is an asymptotic upper bound as well. The time complexity of the actual merging procedure is negligible. Accordingly, the worst-case time complexity of `PLANTm` is $\mathcal{O}(n + k \cdot \log(n) + k^2)$.

Predefining the number of resulting superpixels leads to small variations of `PLANT`'s run time, which is beneficial for real-time applications. Furthermore, the growing of a PLANT can be preempted at any time; `PLANT` does not need to be completed in order to produce a reasonable result, i.e., it can be sufficient if less than the preset number of superpixels is generated. `PLANT` trades off its run time against its precision of results. Hence, `PLANT` is eligible for hard real-time scheduling [53].

3.5 Interim Conclusion

A PLANT is a partition tree that divides data into homogeneous regions. By inserting a partitioning plane, a node splits into further nodes. A binary search determines the plane's location. The binary search examines mean values of features in selected regions; it employs integral images to compute the mean feature values of any region in constant time. If another node is requested to split, a priority queue determines which node splits next—it returns the most heterogeneous of all current leaves.

The superpixel creation algorithm `PLANT` interprets each leaf of a two-dimensional PLANT as a superpixel. Beyond that, the superpixel creation algorithm `PLANTm` merges similar adjacent leaves to obtain superpixels. Both algorithms, but especially `PLANT`, properly comply with most superpixel properties. With a notably small constant factor, `PLANT` and `PLANTm` have order of n time complexity, where n is the number of pixels. Besides, the time complexity of both algorithms depends on the number of resulting superpixels. A detailed evaluation of their run times is given in chapter 5.

[3] However, it is possible to implement balancing for a PLANT if it becomes necessary.

4 Application to Vision of Soccer-playing Robots

PLANT has been employed within the context of the student project FUmanoids in the first place. The PLANT data structure has been developed to lay the groundwork for the computer vision of the soccer-playing robots of the FUmanoids team as described in the team description paper for the RoboCup 2017 competition [23].

4.1 Requirements

The FUmanoids robots compete in the Kid Size class of the Humanoid League. Only human-shaped robots, equipped with human-like sensors, featuring a height between 60 cm and 90 cm are allowed to participate in this competition. The rules of the Humanoid League [43] evolve continuously. In early RoboCup competitions, the rules covered some important elements of soccer merely, in order to establish a flow of play in the matches. As the ambition is to converge to the FIFA's laws of the game, these rules are adjusted through the years.

Recent rule changes have affected the appearance of the ball, of the goals, and of the playing field. Concretely, the robots play on a field consisting of artificial turf. Artificial turf features a variant brightness due to reflections and shadows of the blades of grass. Pixelwise classification of the robot's environment, as described by Rughöft [79], becomes thus more difficult. Consequently, an improved segmentation, considering the spacial environment of the pixels, needs to be done. In addition, instead of playing with a completely red ball, the current rules state that half of the ball's surface area must be colored white, while the remaining area's appearance is undefined.

It is essential for a soccer robot not to confuse the ball with other entities such as field lines or goalposts. For the purpose of a proper classification result, with high precision and recall, a ball candidate should not consist of a too small number of pixels. When using a HD resolution (1280×720 pixels), the robots robustly recognize a ball that is up to approximately 5 m away without any false positive detection. To see the ball across a humanoid soccer field with the dimensions of $9\,\mathrm{m} \times 6\,\mathrm{m}$, the robot's camera has to deliver images in Full HD resolution, which is 1920×1080 pixels. The processing of such a large image constitutes the run-time bottleneck for the robot's cognition. The whole cognition is intended to run with a frame rate of 30 Hz on an ODROID-X2 featuring a 1.7 GHz quad-core CPU. Up to the point where ball candidates and goalposts are detected, the computer vision should not have spent more than 25 ms on computing because it needs to be considered that the remaining parts of the cognition require some time for execution as well. As there is a lack of algorithms that can be implemented to detect the objects that fast, PLANT-based algorithms have been developed to extract the essential information from the stream of high-resolution images.

4.2 Field Contour Extraction

The field contour divides the image into two segments: Field and non-field. Anything outside the field needs not to be considered in subsequent processing steps; by omitting non-field pixels, computation time is saved. In accordance with the laws of the game, the field is an even area of artificial green grass. A field-colored border strip, which has a width of at least 70 cm, surrounds the field. The appearance of the world beyond the border strip is not defined at all. Virtually always, the field is bounded and the color alters behind the border strip; therefore, it can be assumed that the field color is discontinued.

Whether a YCbCr-triplet is classified as a field color mainly depends on the values of the chrominance channels. The field color is almost independent of the luminance. In addition, undesirable reflections and shadows of the lawn are projected onto the Y-channel. Hence, only both chrominance channels are considered to extract the field contour. When a node of a PLANT is split, there are eight distinct cases regarding the arrangement and the classified colors of the resulting nodes: Both of the nodes either feature field color or not, and the split could be made horizontally or vertically.

Initially—when the PLANT consists of a single node, which contains the whole image—the field contour is assumed to be on the bottom border of this node. While the PLANT creates new nodes, the field contour adjusts itself within the range of the corresponding nodes. The field contour ascends to the split location if, firstly, the respective split occurred in vertical direction, secondly, the upper node features no field color, but thirdly, the lower node features field color, and finally, the current field contour is situated below the split location (see Figure 4.1c). Likewise, the field contour ascends to the top of a node if, firstly, it results from a split in horizontal direction, secondly, the respective node features field color, and finally, the current field contour is situated below the top of the respective node (see Figures 4.1g, 4.1h, and 4.1i). As opposed to this, the field contour descends to the bottom of a node if and only if, firstly, it results from a split in horizontal direction, secondly, the respective node features no field color, and finally, the current field contour is current field contour is situated between the top and the bottom of the respective node (see Figures 4.1f, 4.1g, and 4.1h). In all remaining cases,

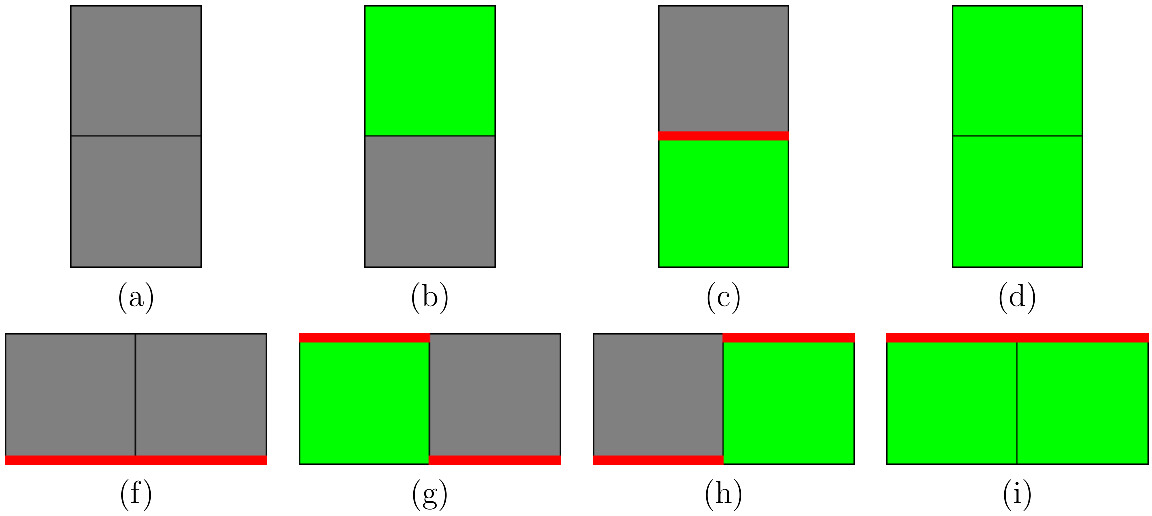


Figure 4.1: Field contour extraction. All eight variants of resulting nodes after splitting are represented. Top: Splits in vertical direction. Bottom: Splits in horizontal direction. Green illustrates that the node is classified as field; gray illustrates that the node is classified as non-field. The red line shows the position where the field contour moves to. The field contour is adjusted to the top of a green node only if it moves the field contour up; likewise, the field contour is adjusted to the bottom of a gray node that resulted from a horizontal split only if the field contour crossed this gray node before. No red line means no adjustment.

including the remaining node arrangements (see Figures 4.1a, 4.1b, and 4.1d), the field contour stays at its position.

As the playing field is constructed on a level ground, nodes that are completely located above the horizon will not split further. By then, the field contour is a sequence of horizontal lines. The center points of these lines add up to interim field contour points. To remove outliers, in the end, the field contour points are smoothed to a convex hull by Andrew's monotone chain algorithm [5], which computes the convex hull efficiently. Figure 4.2 illustrates the intermediate steps and the result of the field contour extraction.

4.3 Object Detection

To detect objects, another PLANT is built. As stated above, the robot considers only the area below its horizon to detect the field. Likewise, it considers only the area below the field contour to detect objects of importance. Accordingly, nodes that are situated above the field contour do not split further. The surface of the ball is defined to contain at least 50 % white color, while the goalposts are defined to be completely white. Consequently, white blobs are detected to generate proposals for these objects. To detect the white blobs, it is sufficient to build a PLANT by considering only the Y-channel of the YCbCr color model. A white blob is detected when a node's mean color is classified as white and its split results in two nodes whose mean colors are white as well (see Figure 4.3). A node that is a part of a white blob does not split further. Similarly, a node whose smallest side projected to the ground is smaller than 5 cm, which can be computed by applying the camera matrix, does not split further.

The result is an image containing non-overlapping bounding boxes of white regions, which can be interpreted as superpixels. These bounding boxes are the basis for the ball and goalpost detection. In this section, the objects of interest are soccer balls and goalposts; the perception of these object is essential for playing soccer. Other objects such as field lines, which are defined to be white, or other robots, which are defined to have black feet and colored jerseys, could be detected in a similar way.

4.3.1 Ball Detection

A detected white blob is an approximate estimation of a potential ball location in the image. Figure 4.4 shows a scenario in which six white blobs are detected, while only one contains the ball. Therefore, these object proposals need to be processed further. At first, the bounding boxes that contain white blobs are trimmed. Sometimes the bounding boxes clearly contain non-white areas as Figure 4.5 shows. The trimming is realized by building another PLANT. This trimming PLANT is built considering the Cb and Cr channels of the YCbCr color model. Using both chrominance channels, especially green

Figure 4.2: Processing steps of the field contour extraction; nodes are displayed in their mean color. Top left: PLANT after first split. The split matches the field contour roughly. Top center: Increased number of nodes. Corner of the field becomes visible. Top right: Increased number of nodes. This segmentation would be sufficient at this point. Bottom left: Final PLANT. The algorithm stops at a certain threshold. Bottom center: Superimposed resulting field contour as a red line. Bottom right: Original image including the field contour.

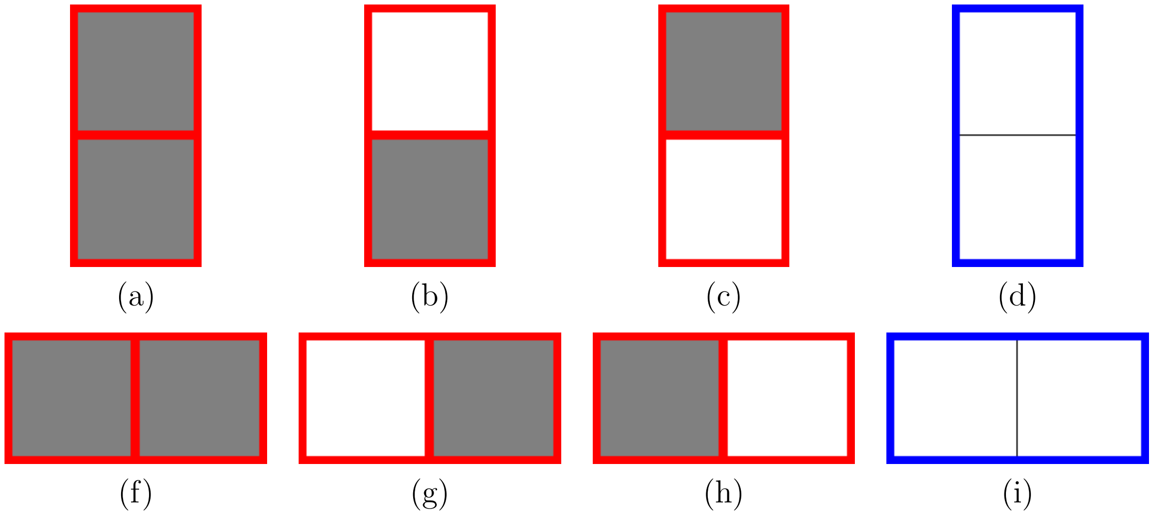


Figure 4.3: White blob detection. All eight variants of resulting nodes after splitting are represented. White illustrates that the mean color of a node is classified as white. Gray illustrates that the mean color of a node is not classified as white. A blue frame is considered as a bounding box of a white blob. A red framed node splits further; in contrast, the nodes that are contained in a blue frame do not split further.

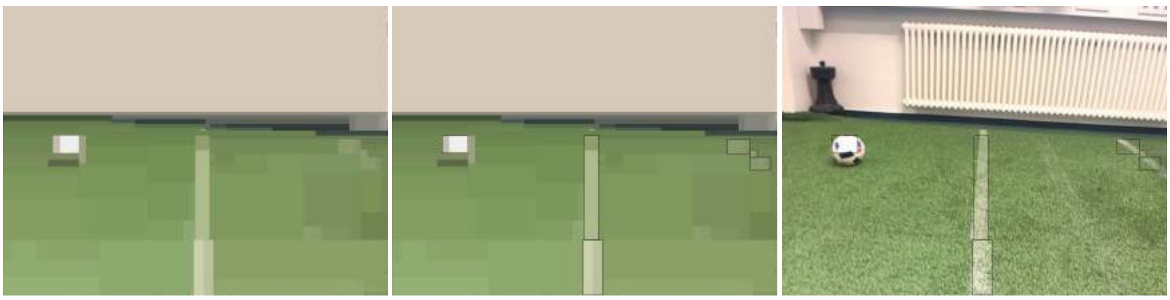


Figure 4.4: White blobs for ball detection. Left: Illustration of PLANT that is built to find white blobs. Center: Black rectangles mark white blobs. Right: Marked white blobs in the original image.

areas are separated well from the ball. While the trimming PLANT is generated, each leave that is not colored white is trimmed (see Figures 4.6a,4.6b, 4.6c, 4.6f, 4.6g, and 4.6h). Additionally, white nodes splitting into two white nodes do not split further and result in an improved bounding box that encloses a white blob (see Figures 4.6d and 4.6i). The trimming PLANT is usually rather small regarding the number of nodes. Unlike the white blob PLANT, the lower bound for the size of a node is one pixel.

Sometimes multiple boxes superimpose the ball. Therefore, boxes that touch each other are merged (see Figure 4.7a). Yet, the merged boxes do not necessarily cover the whole ball. Often only the upper half of the ball is found because it is much brighter than the lower half. This is why, next, the size of the bounding box is adapted to the expected size of the ball. The center point of the box's top side is the fix point while the size is adapted.

Following this, the position of the bounding box is further improved. A fast mode seeking algorithm is implemented that utilizes integral images by comparing average color values of boxes. Mean shift is commonly used for mode seeking [17], but it would be unnecessarily computationally expensive at this point. In the proposed method, the initial bounding box is moved in all four possible directions (vertically up and down, horizontally left and right) by an offset, which is initially set to a quarter of the ball's diameter, to check whether any of these positions fits better. A box fits better if the sum of its average chrominance values is higher. If another positions fits better, the box is moved to that respective position. Subsequently, the offset is halved and the box is moved again. A well fitting bounding box location is found when the offset equals one pixel. Figure 4.7b shows the resulting bounding box. To obtain even better results, this step could be repeated until convergence. Though, this is not necessary in this application; thus, only one iteration is performed.

Eventually, some final checks are done. Firstly, it is checked whether the mean color of the bounding boxes is distinct from the field color. Secondly, it is checked whether the upper half of the bounding box has higher average luminance than the lower half because the upper half of the ball usually reflects more light. Figure 4.7c demonstrates that these two sanity checks can eliminate several bounding boxes. Finally, the Kullback-Leibler (KL) divergence [71] and a support vector machine (SVM) is applied to sort out the remaining false positives. Both the KL divergence and the SVM are comparatively time-consuming. Only a few bounding boxes can be checked within a millisecond. This is acceptable for real-time applications if not many boxes need to be classified, but applying a sliding window would be too computationally expensive. Exactly this is the reason why creating a PLANT for preprossing is necessary: It drops the total run time for the classifiers significantly.

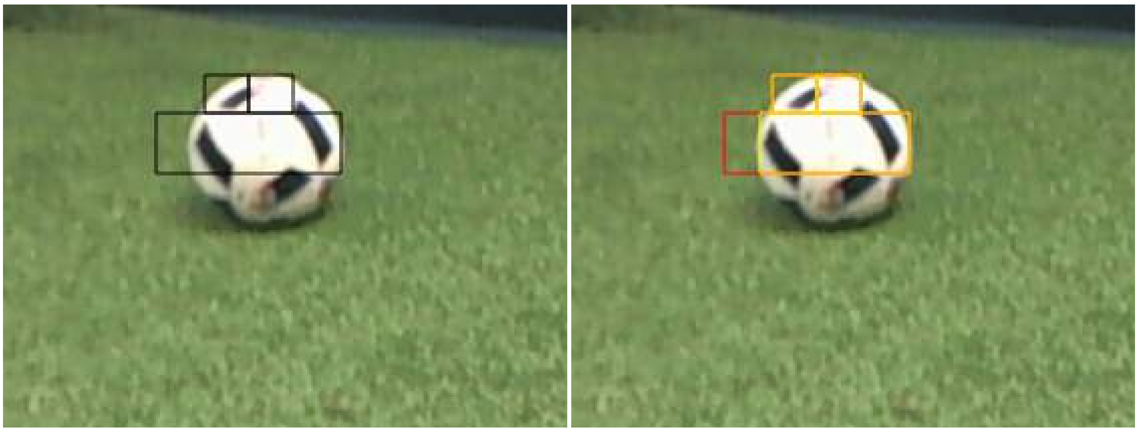


Figure 4.5: Trimming white blob boxes in ball detection (magnified image section). Left: Black bounding boxes contain detected white blobs. Right: Bounding Boxes after trimming are colored yellow; dark yellow for unchanged boundaries and bright yellow for new boundary lines. Red color indicates a part of a bounding box that has been trimmed off. Bright yellow and red lines can be seen on the left of the ball; they enclose the trimmed off area.

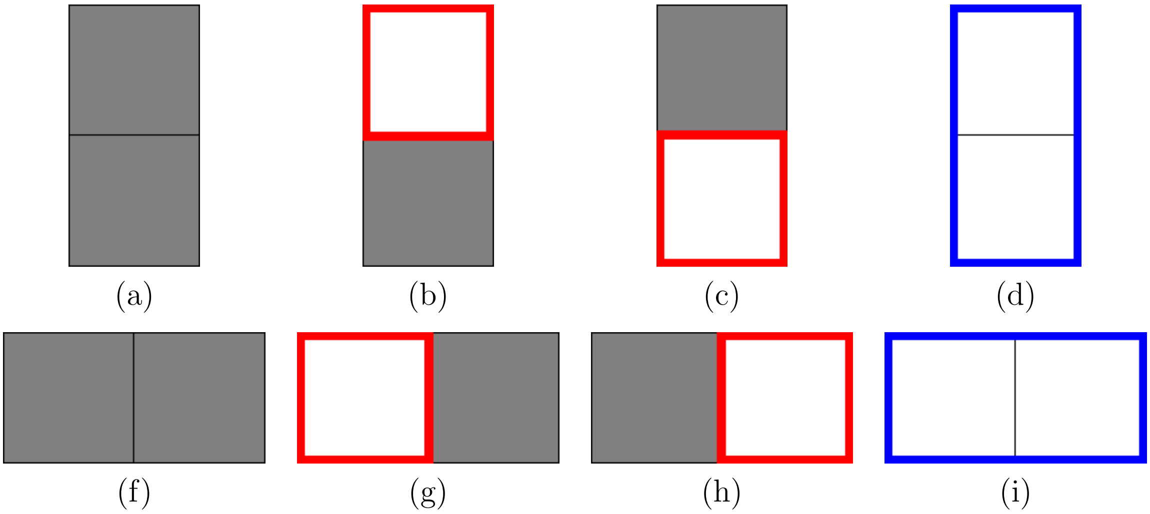


Figure 4.6: Trimming white blobs. All eight variants of resulting nodes after splitting are represented. White illustrates that the mean color of a node is classified as white. Gray illustrates that the mean color of a node is not classified as white. A blue frame is considered as a bounding box of a white blob. A red framed node splits further; in contrast, all other nodes do not split further.

4.3.2 Goalposts Detection

The white blobs are not only used for ball detection, but also for goalposts detection. There are four goalposts on the field. Each goal consists of two goalposts, which have a distance of 260 cm, while both goals are 900 cm apart. Goalposts do not have plenty of features. It is known that they are white and that they have certain dimensions. So, at first, the white blobs are filtered by size (see Figure 4.8). It is checked whether white blobs of the respective dimensions can be a part of a goalpost. However, the size filter is rather permissive in order to not create false negatives. Only boxes that substantially differ in size are eliminated here. Next, the potential goalposts are validated by checking their environment. Additionally, the relative position of the goalposts from the robot are estimated. For these purposes the potential goalposts are scanned up- and downwards.

A goalpost is scanned downwards to locate its relative position on the field. The scanning consists of multiple iterations. Each iteration moves the box downwards by a certain offset (i.e. a quarter of the box's width). Moreover, the same offset is applied to move copies of the downwards shifted box to the left and to the right. Then it is checked which of the three (left, central, right) boxes overlaps the most with a goalpost. For this, the box with the highest average luminance is chosen. This process is iterated until none of the three newly created boxes matches the goal color anymore, and the central box of these three matches the field color. Figure 4.9a shows the initial superpixels and the final boxs touching the field.

The final box contains pixels of both a goalpost and the field. Therefore, also the edge between both is contained in this box. This edge is detected by creating another PLANT within the box. The edge detection PLANT, which considers all three channels of the YCbCr color model, consists of only two leaves. The central point of the separation line is the point where the goalpost touches the field; it is plotted for each goalpost candidate in Figure 4.9b. As this point is situated on the field, the camera matrix can be employed to measure its distance from the robot.

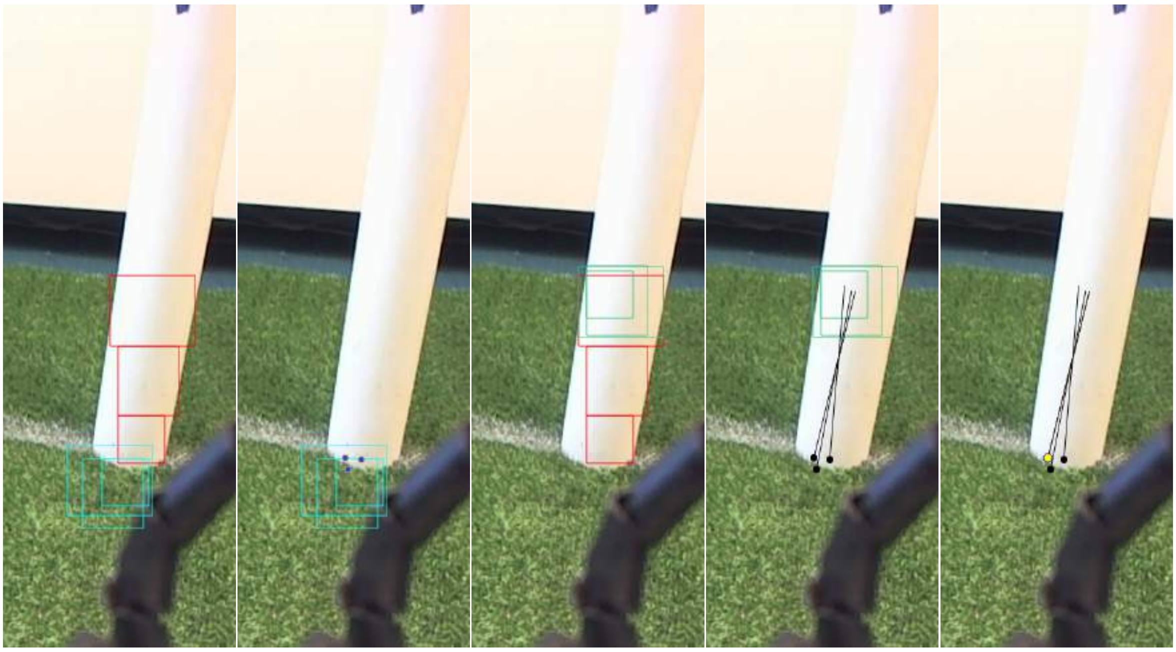
A goalpost is scanned upwards for validation; it is checked whether the goal post crosses the field contour. The upwards scan is done similarly to the downwards scanning described above—but in the other direction. The termination criterion also differs: The scan is stopped as soon as the box touches the field contour. The final boxes of the upwards scan are illustrated in Figure 4.9c. If, by contrast, within an iteration none of the three newly created boxes contains the goal color, the respective superpixels are marked as an invalid goalpost candidate.

Since a goalpost can consist of several superpixels, multiple goalposts can be detected inside the same goalpost. From these detected goalpost candidates, which are true positives, one has to be selected. It is usually negligible which of the candidates is chosen. As a decision has to be made, a candidate originating from a brighter and larger superpixel is preferred. For this, all goalpost candidates that are less than 1 m apart are checked against each other pairwise. The respective goalposts candidates featuring

(a) (b) (c)

Figure 4.7: From bounding boxes containing white blobs to a bounding box that contains the ball. Left: Merging adjacent boxes. Yellow boxes merge to blue boxes. Center: Resizing the boxes to ball size and shift them to non-green regions. Blue boxes transform to magenta boxes. Right: Fast sanity checks to eliminate false positives. All magenta boxes are identified as false positives. The cyan box remains as a valid bounding box.

Figure 4.8: Filter white blobs for goal detection. Left: Illustration of PLANT that is built to find white blobs. Center: Black rectangles mark white blobs. Right: Red rectangles mark potential goalpost superpixel. Black rectangles are eliminated. The left black rectangle is discarded because it is too wide. The right black rectangle is discarded because it is too low.



(a) (b) (c) (d) (e)

Figure 4.9: Extracting a goalpost from white blobs. (a) Red rectangles enclose white blobs. Final rectangles after downwards scanning are colored cyan. (b) Points mark ascertained goalpost positions. (c) Red rectangles enclose white blobs. Final rectangles after upwards scanning are colored cyan. (d) The three detected goalpost candidates are marked by black lines. (e) The yellow marked position is the final position of the goalpost.

the darkest original superpixel are discarded until only one goalpost candidate remains within a radius of 1 m. This candidate is selected as the final goalpost (see Figure 4.9d).

4.4 Interim Conclusion

A PLANT has been used to implement the computer vision for a humanoid soccer-playing robot. Since the nodes of a PLANT split at significant edges, the structure of an image is analyzed while the PLANT is created. Additionally, the PLANT smooths the textured grass by computing the average colors for its nodes; so, the pixel are put into context and outliers are prevented. Instead of using the PLANT to create all superpixels, specialized small PLANTs are generated with the purpose to improve the run time. These specialized PLANTs feature specialized termination criteria and are built on selected color channels. Figures 4.10a and 4.10b illustrate the PLANTs that extract the field contour and detect white blobs. Figure 4.10c shows a summarizing image in which the field contour and a ball in front of two goalposts is detected. Furthermore, this chapter has also presents a fast mode seeking algorithm to improve the positions of bounding boxes. This PLANT-based vision allows the robots to run with a frame rate of 30 Hz at a Full HD resolution. A detailed run time evaluation of a superpixel creating PLANT is given in chapter 5.

(a) (b) (c)

Figure 4.10: Summary of the robot's vision. Left: PLANT to extract field contour. Center: PLANT to extract white blobs. Right: Extracted field contour, ball, and goalposts are marked.

5 Experiments

This chapter evaluates the superpixel creation algorithms `PLANT` and `PLANTm`, which are introduced in section 3.3. They are compared to selected superpixel algorithms presented in section 2.4. The focus is on the evaluation of run times. Additionally, it is examined whether the algorithms produce superpixels of proper quality. All experiment were conducted using the benchmark framework provided by Stutz et al. [88].

5.1 Run Times

Setup

The run times of superpixel creation algorithms were compared on several image resolutions; especially, high resolutions were considered. For lack of data sets of high-resolution images featuring ground truth segmentations, the fundus data set from Köhler et al. [45] was used for efficiency evaluations. The data set is composed of 45 images with a resolution of 3504×2336 pixels. While keeping their aspect ratio, the images were downscaled nine times by 38.2%. Consequently, the image sizes of the resulting 10 different scales varied approximately from 0.1 to 8.2 megapixels. The data set was split into a training set, comprising 12 images, and a test set, comprising 33 images. The training set was used for parameter tuning, whereas the test set was only used for the final experiments. All experiments ran on an Intel Core i5-6200U processor. Two cores, which process data with a clock rate of 2.3 GHz, are attached on its socket. Both cores share an L3 Cache with a capacity of 3 MB.

Experiment I

Method The run times for the six lower image scales—up to a resolution of 1339×891 pixels, which are 1.1 megapixels—were measured. It was attempted to produce 1000 superpixels in all trials. The number of superpixels for `ERS` and `PLANT` was accurately controllable. For the other algorithms, parameters were chosen that result in approximately 1000 superpixels.

Evaluation Figure 5.1a illustrates the run times from a resolution of 402×267 pixels up to a resolution of 1339×891 pixels. It displays that `PLANT` has the lowest processing time on all image scales. `PLANTm` is faster than all non-PLANT-based algorithms for resolutions that are higher than approximately 0.25 megapixels.

Experiment II

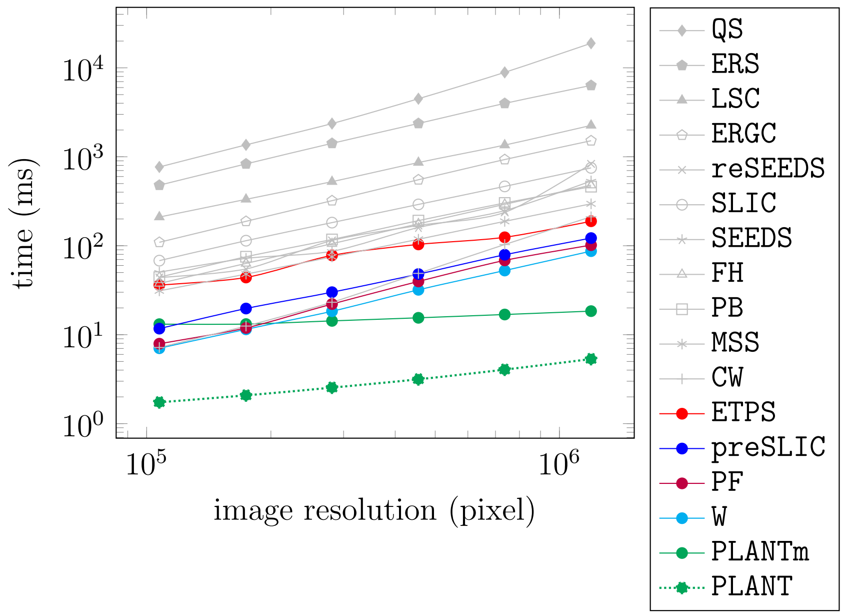
Method For some algorithms, their parameters can be tuned to reach a faster run time. Thus, the parameters were adapted to produce 1000 superpixels with a minimum run time. In contrast to Experiment I, additional parameters that do not affect the number of superpixels were adapted, which potentially lowers the quality of the resulting superpixels. This experiment was conducted on the same data as Experiment I. Minimizing the run time of `PLANTm` would result in the `PLANT` algorithm, which applies no merging at all; thus, `PLANT` can be considered as a fast version of `PLANTm`. As `PLANT` had already been evaluated in Experiment I and cannot be optimized further by adjusting parameters, `PLANT` and `PLANTm` are not considered in this experiment.

Evaluation The run times for the original and fast versions of the algorithms are shown in Figure 5.1b. The fast versions of the algorithm are denoted by attaching a `f` to the acronyms. Figure 5.1b illustrates that especially `preSLICf`, `SLICf` and `SEEDSf` achieved low run times.

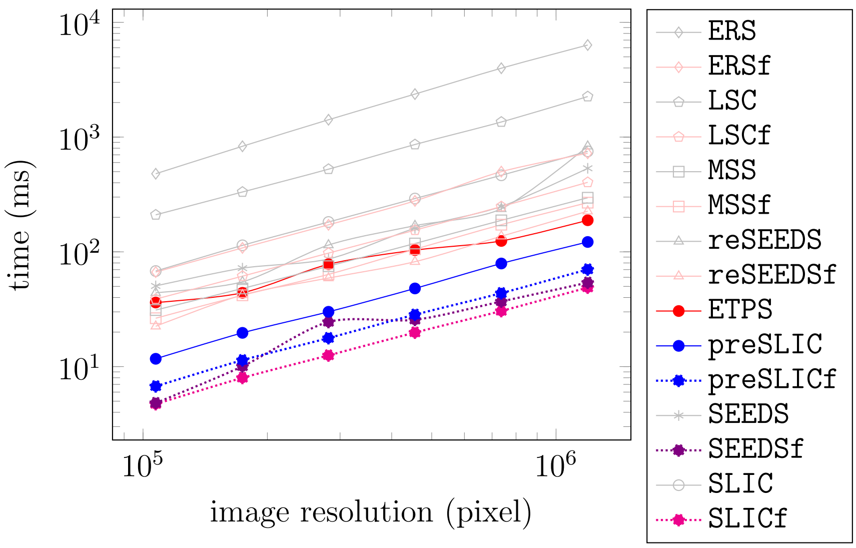
Experiment III

Method This experiment considers the run times at resolutions up to 8.2 megapixels (see Figure 5.2a). `ETPS` has been ranked as the top performing algorithm in the superpixel benchmark [88]. For this experiment, it was therefore assumed that all other superpixel algorithms generate superpixels of lower quality. Hence, algorithms whose both run time and slope of run time were higher than those of `ETPS` at a resolution of 1339×891 pixels were not considered for this experiment. These algorithms are shown grayed out in Figures 5.1a and 5.1b.

Evaluation `PLANT` and `PLANTm` were the only superpixel algorithms running with a frame rate of at least a 30 Hz on images with at least HD resolution, i.e. 1280×720 pixels (see Figure 5.2a). Nevertheless, `PLANTm` failed to reach a frame rate of 30 Hz at the full resolution. The fastest non-PLANT-based algorithm was `SLICf` with 368 ms. The slope of `PLANT` and `PLANTm` were significantly lower than the slope of the other superpixel algorithms (see Figure 5.2b). Although the run times of `PLANT` and `PLANTm` are asymptotic linear, in the range of the examined image resolutions, the run times still appeared to increase in a sub-linear manner as the graphs for `PLANT` and `PLANTm` in Figure 5.2c are almost a line with a minor kink. This motivated Experiment IV.

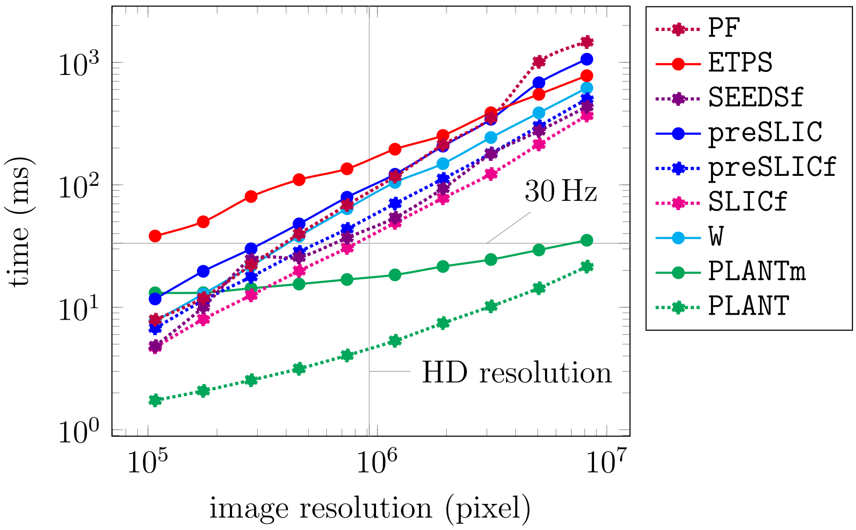


(a) Run times of superpixel algorithms for producing 1000 superpixels on lower resolutions. Algorithms that are slower than `ETPS` are not consider in subsequent run-time experiments. `PLANT` is the fastest algorithm on all resolutions.

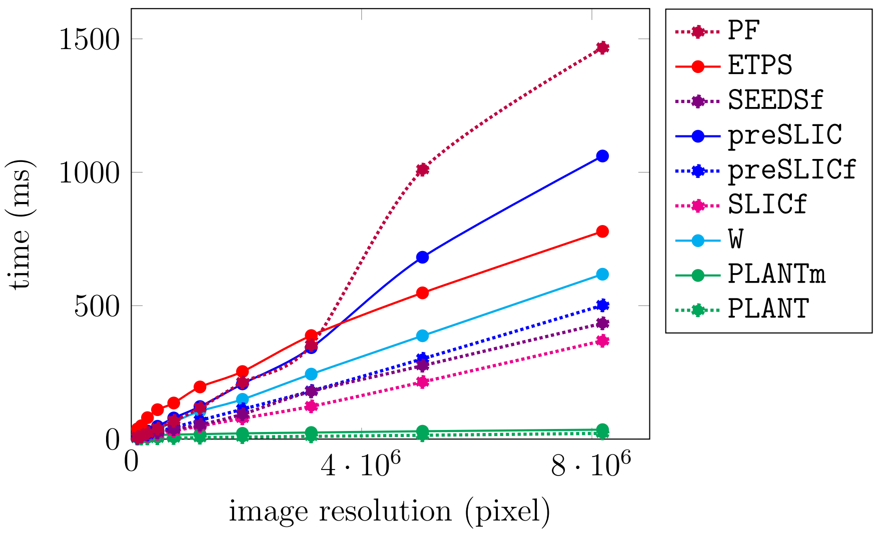


(b) Some Algorithms can be sped up by choosing different parameters. Fast Algorithms that are faster than `ETPS` are further considered. This is true for `SLICf`, `preSLICf` and `SEEDSf`.

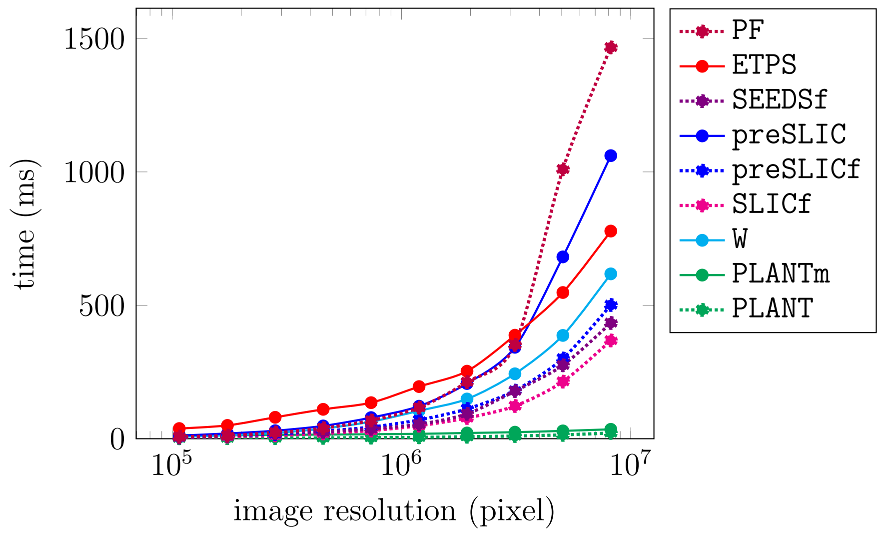
Figure 5.1: Experiments I and II



(a) This illustrates the run time of fast superpixel algorithm on a broad range of resolutions. No algorithm but `PLANT` and `PLANTm` were able to produce superpixels on HD images with a frequency of 30 Hz on the experimental setup.



(b) The linearly scaled graph represents the same data as Figure 5.2a. It illustrates that the slope of the run time over the resolution is much smaller for `PLANT` and `PLANTm` than for the other algorithms. They scale significantly better.



(c) The log-linear graph for `PLANT` and `PLANTm` looks almost like a straight line, which indicates constant or logarithmic behavior. A small kink can be seen starting at around $3 \cdot 10^6$, which indicates a transition into linear behavior.

Figure 5.2: Experiment III

Experiment IV

Method The run times were measured for the different components of `PLANT` and `PLANTm` (see figure 5.3b). The components for `PLANT` are: Firstly, the conversion of an image to an integral image, and secondly, the generation of a PLANT. In addition, `PLANTm` involves the merging procedure as a third step.

Evaluation `PLANT` needed the same time for the integral image conversion and for generating the PLANT at approximately 1.5 million pixels. For higher resolutions, the PLANT creation was faster than the integral image conversion. More precisely, for an image with the dimensions of 1339×891 pixels, both the conversion and the generation required nearly 3 ms; the total time was 5.5 ms. `PLANTm` had the same conversion time because the conversion only depends on the image size. Its generation time was nearly 5 ms as more superpixels were produced for subsequent merging. The merging itself required nearly 12 ms, resulting in a total time of around 19 ms. On the full resolution of 3504×2336, the conversion time was much higher. `PLANT` and `PLANTm` required 17 ms. By contrast, the generation time was only slightly higher—less than 4 ms for `PLANT` and almost exactly 6 ms for `PLANTm`. `PLANTm` required nearly 12 ms for merging. As a result, the total time for `PLANT` and `PLANTm` were 21 ms and 35 ms, respectively. Figure 5.3a and 5.3b illustrate that the conversion behaves linearly and the PLANT generation behaves logarithmically. The run time of the merging procedure does not depend on the number of pixels; thus, it has roughly constant behavior. In summary, the asymptotic most complex term, which is the conversion to an integral image, did not become prevalent until the image is composed of a vast number of pixels.

5.2 Superpixel Quality

Metrics

To compare the quality of different superpixel algorithms, a quantitative evaluation was conducted. The metrics boundary recall, undersegmentation error, and explained variation are used in most superpixel studies. Furthermore, they are expressive and do not correlate much more than necessary [88]. However, it is difficult to quantify solutions of the ill-posed segmentation problem. Therefore, the metrics can only provide a rough estimation of the superpixel quality.

Boundary Recall Boundary recall (Rec) measures the fraction of detected boundary pixels from the ground truth data [60]:

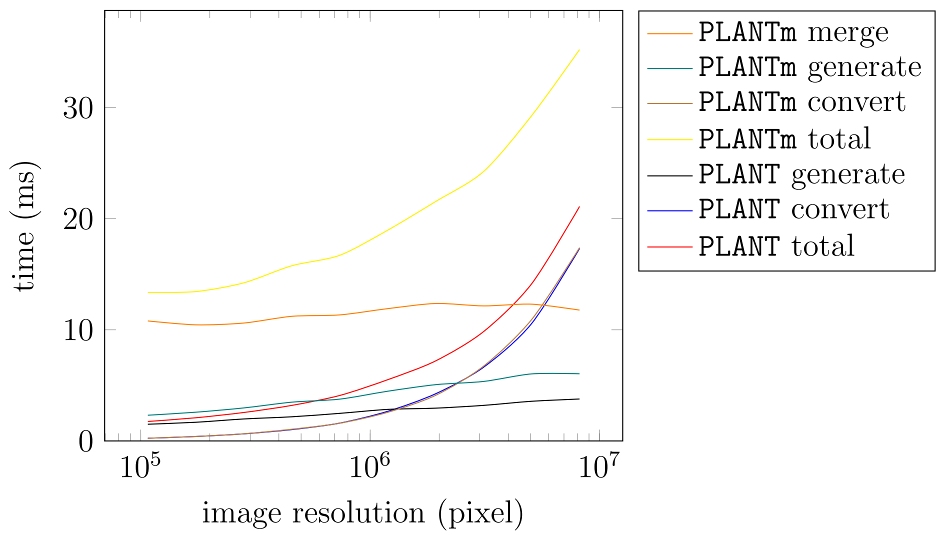
$$Rec = \frac{TP}{TP + FN} \tag{5.1}$$

A ground truth boundary pixel is regarded as detected if a pixel in a predefined, small local neighborhood is classified as boundary—it is a true positive (TP). Every ground truth boundary pixel that remains undetected is a false negative (FN). High-quality superpixel segmentations achieve a high boundary recall since they contain superpixels that adhere to many ground truth boundaries.

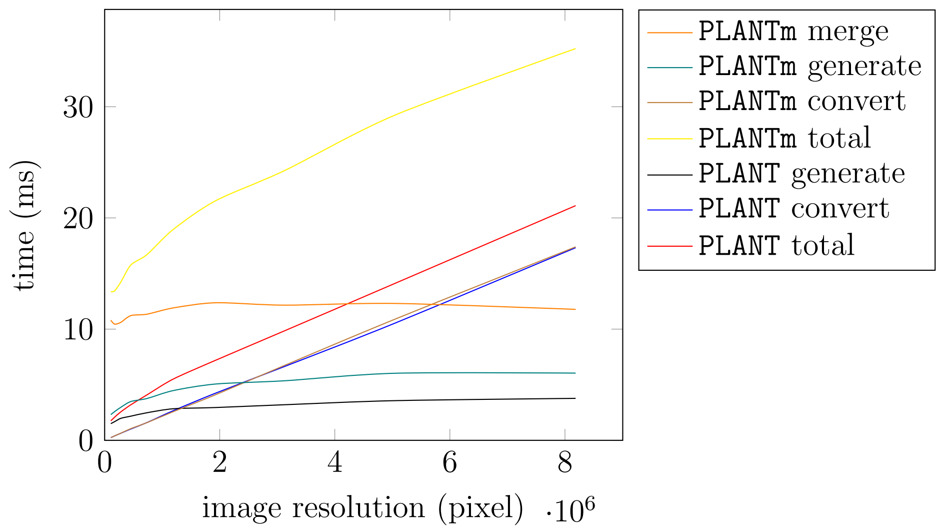
Undersegmentation Error Each pixel of a superpixel should belong to the same object in the real world. The undersegmentatiton error (UE) represents the quantity of pixels that leak into another ground truth segment. There are several formulae for its calculation, this thesis uses the formula defined by Neubert and Protzel [68]:

$$UE = \frac{1}{N} \sum_{G_j \in G} \sum_{P_i \in P} \min(|P_i \cap G_j|, |P_i \setminus G_j|) \tag{5.2}$$

G consists of all ground truth superpixels, P consists of the computed superpixels, and N is the total number of pixels. As a result, the number of pixels that cross a boundary of the ground truth segmentation are counted and divided by the total number of pixels. High-quality superpixel segmentations achieve a low undersegmentation error since their superpixels are rarely part of more than one object.



(a) `PLANT` consists of a convert part to compute the integral image and a build part to generated the tree. In `PLANTm` a merge part follows. It can be seen that building has logarithmic run time and merge is highly independent of number of pixels.



(b) The same data in a linearly scaled graph. It illustrates that the convert time is linear. The linear convert time becomes predominant at around 1.5 million pixels for `PLANT`. For `PLANTm` it becomes predominant at around 4 million pixels.

Figure 5.3: Experiment IV

Explained Variation The explained variation (R^2) measures the variation of the image that is explained by the superpixels if they are represented by their mean color. It is calculated as follows:

$$R^2 = \frac{\sum_i (\mu_i - \mu)^2}{\sum_j (x_j - \mu)^2} \tag{5.3}$$

μ_i is the mean color of the i-th superpixel, μ is the mean color of the image, and x_j is the pixel value of the i-th pixel. High-quality superpixel segmentations achieve a high explained variation since they can represent most information of the image by their superpixels.

Experiment V

Setup and Method The segmentation quality of the superpixel algorithms were evaluated on the Berkley Segmentation Data Set 500 (BSDS500) [61]. It is the commonly used for superpixel evaluation and also segmentation evaluation in general. BSDS500 is composed of outdoor scenes. The data set comprises a test set of 200 images, a training set of 200 images, and a validation set of 100 images. Their resolution is 481×321 pixels. Each image features between four and eight ground truth segmentations, which were produced by humans. All human segmentations should be reproducible by using superpixels as the elementary element. Accordingly, the ground truth segmentation that scores worst was regarded for all metrics. These worst score were averaged over the images.

The parameters of the algorithms were tuned on the training set to produce 300, 1500, and 7500 superpixels with a high boundary recall and a low undersegmentation error. The objection function maximized the difference between both. Consequently, three parameter sets were obtained. Subsequently, the parameters were linearly scaled between the first and the second and the second and the third parameter set to produce additional data points (similar to the benchmark [88]).

Evaluation Figure 5.4 illustrates the results of this experiment. It shows that `PLANTm` expectedly performed slightly better than `PLANT`. The quality of superpixels produced by `PLANTm` and `PLANT`, which is illustrated in Figure 5.5, was similar to the quality of superpixels produced by `W`. It was confirmed that `ETPS` is currently the top-performing superpixels algorithm. `SLIC` and its variations (`SLICf`, `preSLIC`, `preSLICf`) performed better than `PLANT` and `PLANTm`, whereas `SEEDSf` and `PF` performed worse than `PLANT` and `PLANTm`.

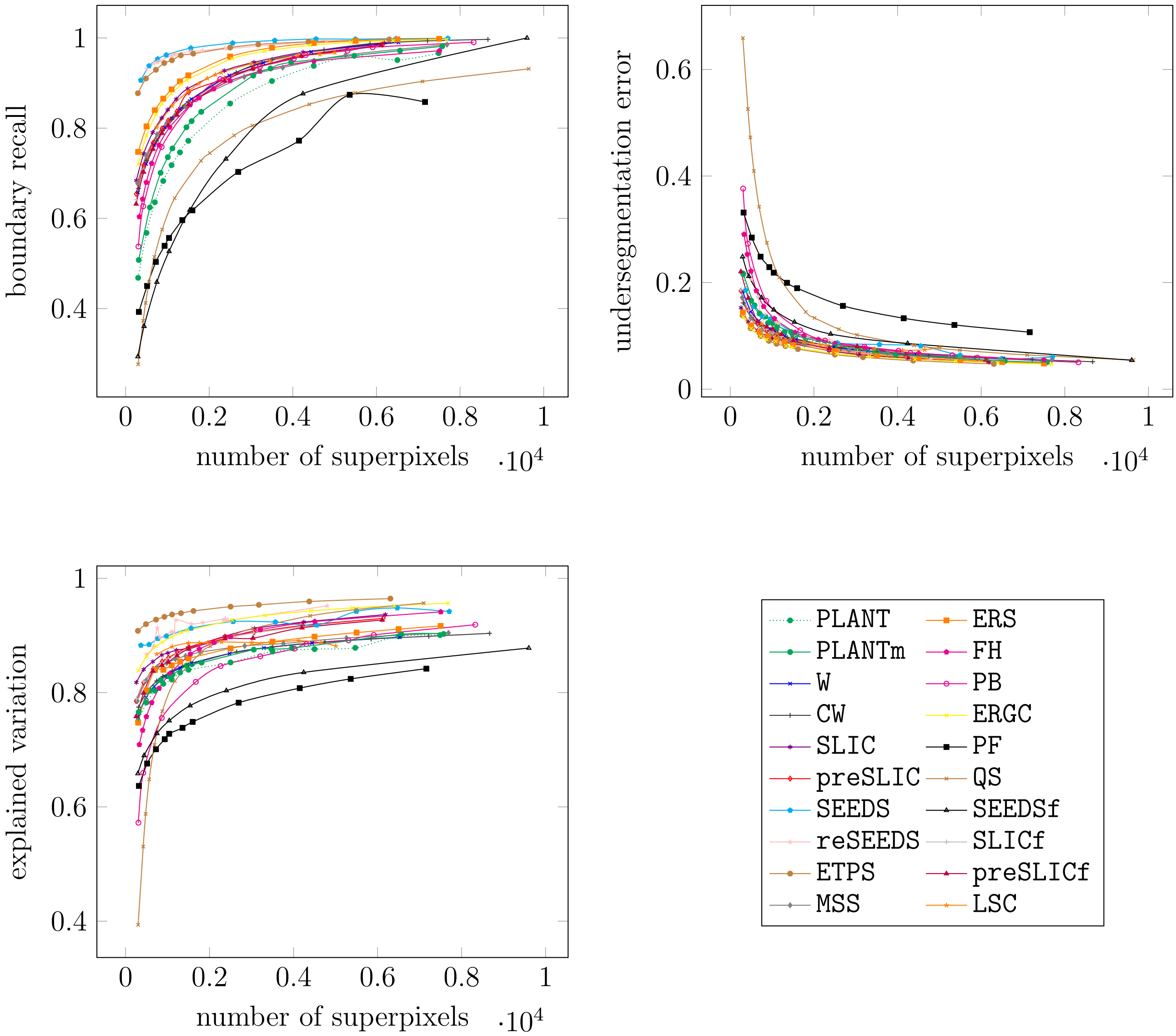


Figure 5.4: Experiment V: It can be seen that `PLANT` and `PLANTm` perform in the same order of magnitude as other superpixel algorithms. It can be seen that it ranks similar to watershed algorithms, while most `SLIC`- and `SEEDS`-based algorithms perform better and `PF` and `SEEDSf` perform worse.

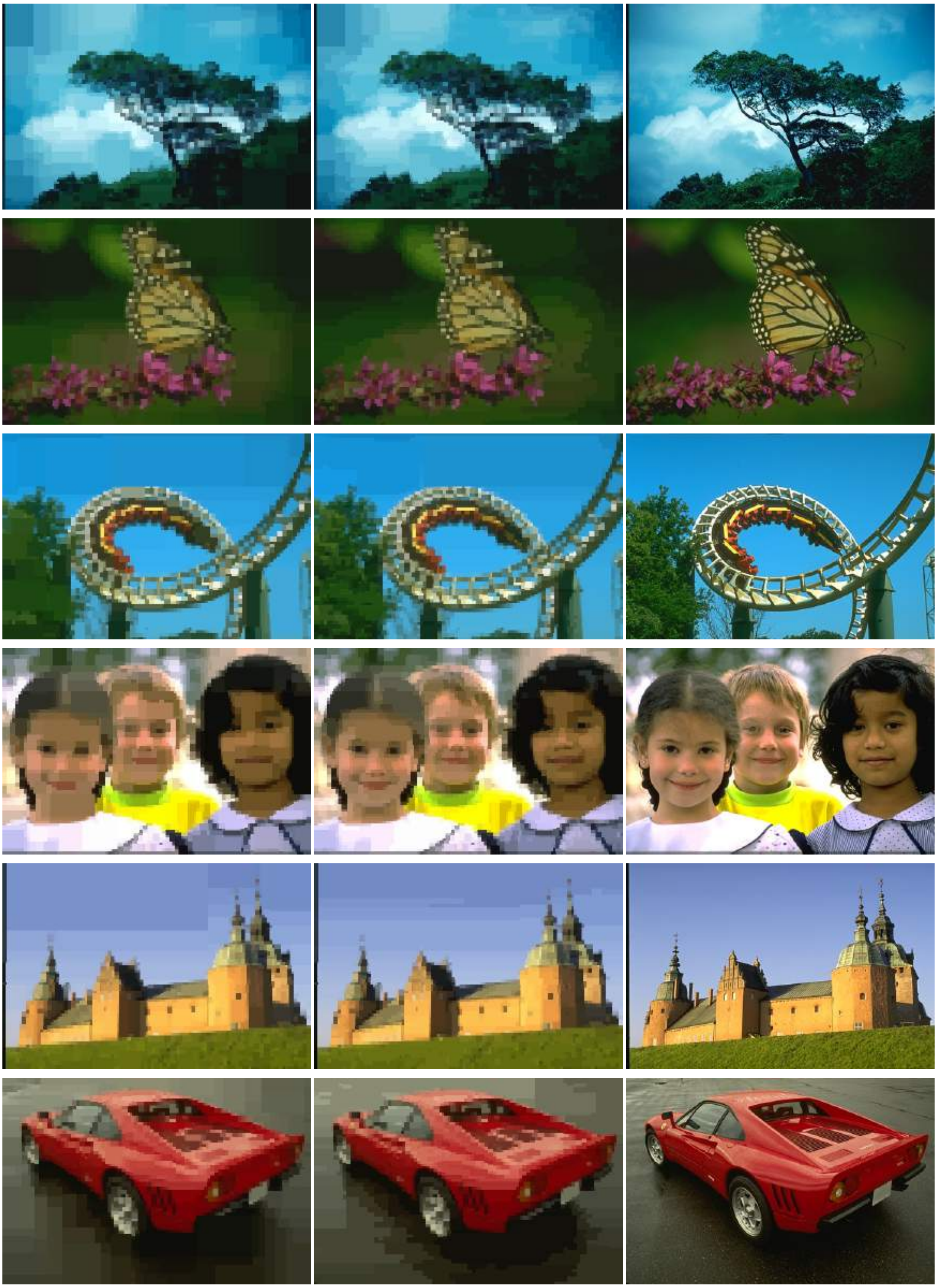


Figure 5.5: From superpixels reconstructed images. Each superpixel is represented by its mean color. All images are part of Berkley Segmentation Dataset 500 [6]. Left: 1500 `PLANT` superpixels. Center: 1500 `PLANTm` superpixels. Right: Original images.

5.3 Interim Conclusion

PLANT and PLANTm are efficient superpixel creation algorithms. Applied on high-resolution images, they are more than an order of magnitude faster than other fast state-of-the-art superpixel creation algorithms. The conversion of an image to an integral image is the asymptotic most complex part of the algorithm; yet the conversion's absolute time is low at the regarded image resolutions. The quality of the superpixels produces by PLANT and PLANTm represents the average of state-of-the-art algorithms. Both algorithms perform similar to W.

Suppose superpixels has to be produces for a large image in short time, it is recommendable to use PLANTm. Compared to other superpixel creation algorithms, it provides a significant speed-up. If the resulting run time is not sufficient, PLANT can be employed. It enables a further noticeable speed-up without forfeiting much superpixel quality. Provided that a higher superpixel quality is required, while the superpixels still have to be created fast, linear iterative clustering methods (SLIC, SLICf, preSLIC, preSLICf) are appropriate algorithms. If the superpixel quality is prioritized, the use of ETPS is advisable. ETPS has a higher run time than the algorithms mentioned above, but it is not inefficient; it supports frame rates higher than 1 Hz for superpixel creation on high resolutions.

6 Conclusion and Future Work

6.1 Conclusion

A PLANT is a binary space partitioning tree. Its leaves encompass homogeneous regions. To achieve real-time performance, a PLANT uses a precomputed integral image to partition an image. Once the integral image is computed, the PLANT can efficiently create nodes through a binary search over image regions of decreasing sizes. The superpixel creation algorithm `PLANT` considers each leave of a PLANT as a superpixel, whereas the superpixel creation algorithm `PLANTm` merges leaves to obtain superpixels. `PLANT` and `PLANTm` comply with characteristical superpixel properties; in particular, both are designed to be efficient.

`PLANT` and `PLANTm` are notably efficient superpixel creation algorithms. In the experiments, both were more than an order of magnitude faster than state-of-the-art algorithms on high-resolution images. They were the only algorithms that could produce superpixels on images containing more than one million pixels with a frame rate of at least 30 Hz.

The quality of superpixels created by `PLANT` and `PLANTm` is comparable to state-of-the-art algorithms; they performs similar to the watershed segmentation algorithm, while being distinctly faster, especially at high-resolutions. If superpixel quality is the priority but efficiency is still not insignificant, the use `SLIC` or `ETPS` is recommended. However, each superpixel creation algorithm produces superpixels that feature different properties, which should also be regarded.

A PLANT was applied to implement a computer vision for soccer-playing robots. While the PLANT for the vision is generated, nodes only split further if the split is expected to provide useful information. The PLANT uses the mean colors, which are low-level cues, of image regions to detect object candidates. After that, higher-level cues are considered to complete the object detection, e.g. to verify a goalpost candidate, its context is regarded. PLANTs enable the robot to use a higher image resolution for their vision. Therefore, the robots obtain more information about their environment they are interacting with, which improves their capabilities.

6.2 Future Work

- A PLANT can be used as a part of a calibration-free vision or to bootstrap an automatic camera calibration. A PLANT produces more leaves at a certain threshold the more information the image provides, which means the parameter is better adjusted. Parameters such as white balance and exposure can be tuned with this method but not parameters such as contrast as it produces noise when adjusted wrongly.
- Field lines and obstacle such as team-mates and opponent players can be detected with a PLANT as well by searching blobs in the determined colors. Field lines are white, and soccer robots have black feet and wear a jersey that is colored in a predefined color.
- An enhanced merging procedure for `PLANTm` should be considered. A better elaborated merging could provide improved results and be more efficient.
- There are still several options to optimize the algorithm and its implementation. For example, even if the chrominance values are subsampled in the input image, the integral images for the chrominance channels are generated in the same size as the integral image for the luminance channel. If the chrominance integral images were generated in the same size as the chrominance input images, the number of write operation would be reduced, which would lead to a speed-up of the integral image conversion.

Bibliography

[1] Radhakrishna Achanta, Francisco Estrada, Patricia Wils, and Sabine Süsstrunk. Salient region detection and segmentation. *Computer Vision Systems*, pages 66–75, 2008.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[3] Alper Aksac, Tansel Ozyer, and Reda Alhajj. Complex networks driven salient region detection based on superpixel segmentation. *Pattern Recognition*, 2017.

[4] Julien Allali, Remi Fabre, Hugo Gimbert, Loic Gondry, Ludovic Hofer, Olivier Ly, Steve N'Guyen, Gregoire Passault, Antoine Pirrone, and Quentin Rouxel. Rhoban Football Club – team description paper – Humanoid Kid-Size League, Robocup 2017 Nagoya. In *RoboCup 2017: Robot Soccer World Cup XXI Preproceedings*. RoboCup Federation, Forthcoming 2017.

[5] Alex M Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979.

[6] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011.

[7] Pablo Arbeláez, Jordi Pont-Tuset, Jonathan T Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014.

[8] Alper Ayvaci and Stefano Soatto. Motion segmentation with occlusions on the superpixel graph. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 727–734. IEEE, 2009.

[9] Wanda Benesova and Michal Kottman. Fast superpixel segmentation using morphological processing. In *Proceedinks of the International Conference on Machine Vision and Machine Learning-MVML 2014*, 2014.

[10] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[11] Robert Birkus. Accelerated gSLIC for superpixel generation used in object segmentation. In *Proc. of CESCG'15*, 2015.

[12] András Bódis-Szomorú, Hayko Riemenschneider, and Luc Van Gool. Superpixel meshes for fast edge-preserving surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2011–2020, 2015.

[13] Derek Bradley and Gerhard Roth. Adaptive thresholding using the integral image. *Journal of graphics, gpu, and game tools*, 12(2):13–21, 2007.

[14] Pierre Buyssens, Isabelle Gardin, and Su Ruan. Eikonal based region growing for superpixels generation: Application to semi-supervised real time organ segmentation in CT images. *IRBM*, 35(1):20–26, 2014.

[15] Peter Carr and Richard Hartley. Minimizing energy functions on 4-connected lattices using elimination. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2042–2049. IEEE, 2009.

[16] Jun Cheng, Jiang Liu, Yanwu Xu, Fengshou Yin, Damon Wing Kee Wong, Ngan-Meng Tan, Dacheng Tao, Ching-Yu Cheng, Tin Aung, and Tien Yin Wong. Superpixel classification based optic disc and optic cup segmentation for glaucoma screening. *IEEE Transactions on Medical Imaging*, 32(6):1019–1032, 2013.

[17] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, 17(8):790–799, 1995.

[18] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[19] Timothee Cour, Florence Benezit, and Jianbo Shi. Spectral segmentation with multiscale graph decomposition. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1124–1131. IEEE, 2005.

[20] Franklin C Crow. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, 18(3):207–212, 1984.

[21] Pablo De Cristóforis, Matías A Nitsche, Tomáš Krajník, and Marta Mejail. Real-time monocular image-based path detection. *Journal of Real-Time Image Processing*, 11(2):335–348, 2016.

[22] Xingping Dong, Jianbing Shen, Ling Shao, and Luc Van Gool. Sub-markov random walk for image segmentation. *IEEE Transactions on Image Processing*, 25(2):516–527, 2016.

[23] Jan Draegert, Simon Gene Gottlieb, Michael Pluhatsch, Arne Schmidt, Till-Julius Krüger, Anahid Roshandel, Christopher Mühl, and Raúl Rojas. Berlin United - FUmanoids team description paper for RoboCup 2017. In *RoboCup 2017: Robot Soccer World Cup XXI Preproceedings*. RoboCup Federation, Forthcoming 2017.

[24] Fabio Drucker and John MacCormick. Fast superpixels for video analysis. In *Motion and Video Computing, 2009. WMVC'09. Workshop on*, pages 1–8. IEEE, 2009.

[25] Anders P Eriksson, Carl Olsson, and Fredrik Kahl. Normalized cuts revisited: A reformulation for segmentation with linear grouping constraints. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[26] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[27] Hafez Farazi, Philipp Allgeuer, and Sven Behnke. A monocular vision system for playing soccer in low color information environments. In *Proceedings of 10th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conference on Humanoid Robots, Seoul, Korea*, 2015.

[28] G Th Fechner. *Elemente der psychophysik*. 1860.

[29] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[30] Gang Fu, Hongrui Zhao, Cong Li, and Limei Shi. Segmentation for high-resolution optical remote sensing imagery using improved quadtree and region adjacency graph technique. *Remote sensing*, 5(7):3259–3279, 2013.

[31] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, volume 14 of *SIGGRAPH '80*, pages 124–133, New York, NY, USA, 1980. ACM.

[32] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[33] Roberto Gilmozzi and Jason Spyromilio. The european extremely large telescope (e-elt). *The Messenger*, 127(11):3, 2007.

[34] Daniela Giordano, Francesca Murabito, Simone Palazzo, and Concetto Spampinato. Superpixel-based video object segmentation using perceptual organization and location prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4814–4822, 2015.

[35] Rémi Giraud, Vinh-Thong Ta, and Nicolas Papadakis. Robust shape regularity criteria for superpixel evaluation. *IEEE International Conference on Image Processing*, 2017.

[36] Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan, and Daphne Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008.

[37] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Indoor scene understanding with RGB-D images: Bottom-up segmentation, object detection and semantic segmentation. *International Journal of Computer Vision*, 112(2):133–149, 2015.

[38] Eric Hamilton. JPEG file interchange format. Technical report, C-Cube Microsystems, Milpitas, CA, USA, 1992.

[39] Shengfeng He, Rynson WH Lau, Wenxi Liu, Zhe Huang, and Qingxiong Yang. SuperCNN: A superpixelwise convolutional neural network for salient object detection. *International Journal of Computer Vision*, 115(3):330–344, 2015.

[40] Xuming He, Richard Zemel, and Debajyoti Ray. Learning and incorporating top-down cues in image segmentation. *Computer Vision–ECCV 2006*, pages 338–351, 2006.

[41] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, 2005.

[42] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using RGB-D cameras. In *Robot Soccer World Cup*, pages 306–317. Springer, 2011.

[43] Humanoid League Technical Committee of RoboCup 2017. *RoboCup Soccer Humanoid League Laws of the Game 2016/2017*, March 2017.

[44] ITU-R. Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. ITU-R Recommendation BT.601-7, 2011.

[45] Thomas Köhler, Attila Budai, Martin F Kraus, Jan Odstrčilik, Georg Michelson, and Joachim Hornegger. Automatic no-reference quality assessment for retinal fundus images using vessel segmentation. In *Computer-Based Medical Systems (CBMS), 2013 IEEE 26th International Symposium on*, pages 95–100. IEEE, 2013.

[46] Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, Sven J. Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297, 2009.

[47] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. In *ACM Transactions on Graphics (ToG)*, volume 23, pages 303–308. ACM, 2004.

[48] Zhengqin Li and Jiansheng Chen. Superpixel segmentation using linear spectral clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1356–1363, 2015.

[49] Zhenguo Li, Xiao-Ming Wu, and Shih-Fu Chang. Segmentation using superpixels: A bipartite graph partitioning approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 789–796. IEEE, 2012.

[50] Dahua Lin, Sanja Fidler, and Raquel Urtasun. Holistic scene understanding for 3D object detection with RGBD cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1417–1424, 2013.

[51] Bin Liu, Hao Hu, Huanyu Wang, Kaizhi Wang, Xingzhao Liu, and Wenxian Yu. Superpixel-based classification with an adaptive number of classes for polarimetric SAR images. *IEEE Transactions on Geoscience and Remote Sensing*, 51(2):907–924, 2013.

[52] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[53] Jane W-S Liu, K-J Lin, W-K Shih, J-Y Chung, W Zhao, et al. Algorithms for scheduling imprecise computations. *Computer*, 24(5):58–68, 1991.

[54] Miaomiao Liu, Mathieu Salzmann, and Xuming He. Discrete-continuous depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723, 2014.

[55] Ming-Yu Liu, Oncel Tuzel, Srikumar Ramalingam, and Rama Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2097–2104. IEEE, 2011.

[56] Si Liu, Jiashi Feng, Csaba Domokos, Hui Xu, Junshi Huang, Zhenzhen Hu, and Shuicheng Yan. Fashion parsing with weak color-category labels. *IEEE Transactions on Multimedia*, 16(1):253–265, 2014.

[57] Miriam Lopez-de-la Calleja, Takayuki Nagai, Muhammad Attamimi, Mariko Nakano-Miyatake, Hector Perez-Meana, et al. Object detection using SURF and superpixels. *Journal of Software Engineering and Applications*, 6(09):511, 2013.

[58] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1854–1861, 2013.

[59] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Vincent Lepetit, and Pascal Fua. A fully automated approach to segmentation of irregularly shaped cellular structures in EM images. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*, pages 463–471, 2010.

[60] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

[61] David R. Martin, Charless C. Fowlkes, D. Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[62] Heinrich Mellmann, Benjamin Schlotter, Steffen Kaden, Philipp Strobel, Thomas Krause, and Claas-Norman Ritter. Berlin United - Nao Team Humboldt: Team report 2016. Technical report, Humboldt-Universität zu Berlin, Adaptive Systems Group, 2016.

[63] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3061–3070, 2015.

[64] Fernand Meyer. Color image segmentation. In *Image Processing and its Applications, 1992., International Conference on*, pages 303–306. IET, 1992.

[65] Hossein Mobahi, Shankar R Rao, Allen Y Yang, Shankar S Sastry, and Yi Ma. Segmentation of natural images by texture and boundary compression. *International journal of computer vision*, 95(1):86–98, 2011.

[66] Alastair P. Moore, Simon J. D. Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[67] Greg Mori, Xiaofeng Ren, Alexei A Efros, and Jitendra Malik. Recovering human body configurations: Combining segmentation and recognition. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–II. IEEE, 2004.

[68] Peer Neubert and Peter Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, pages 1–12, 2012.

[69] Peer Neubert and Peter Protzel. Compact watershed and preemptive SLIC: On improving trade-offs of superpixel segmentation algorithms. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 996–1001. IEEE, 2014.

[70] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 733–740. IEEE, 2012.

[71] Michael Pluhatsch. Objekterkennung anhand farbklassifizierter integralbilder und kullback-leibler-divergenz. Bachelor thesis, Freie Universität Berlin, Forthcoming 2017.

[72] Todd M Preuss and Ghislaine Q Coleman. Human-specific organization of primary visual cortex: alternating compartments of dense Cat-301 and calbindin immunoreactivity in layer 4A. *Cerebral Cortex*, 12(7):671–691, 2002.

[73] Yao Qin, Huchuan Lu, Yiqun Xu, and He Wang. Saliency detection via cellular automata. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 110–119, 2015.

[74] Sepehr Ramezani, Amirali Setaieshi, Nima Pourmohammadi, Parsa Yarahmadi, Armin Arvand, Foruzan Fallah, Amirhossein Hosseinmemar, Jamillo Santos, Kyle Morris, Meng Cheng Lau, John Anderson Jacky Baltes, and Sourosh Sadeghnejad. AUTMan humanoid teen size team description paper RoboCup 2017 humanoid robot league, Nagoya, Japan. In *RoboCup 2017: Robot Soccer World Cup XXI Preproceedings*. RoboCup Federation, Forthcoming 2017.

[75] Pekka Rantalankila, Juho Kannala, and Esa Rahtu. Generating object segmentation proposals using global and local search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2417–2424, 2014.

[76] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *ICCV*, volume 1, pages 10–17, 2003.

[77] Joris Roels, Jonas De Vylder, Yvan Saeys, Bart Goossens, and Wilfried Philips. Decreasing time consumption of microscopy image segmentation through parallel processing on the GPU. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 147–159. Springer, 2016.

[78] Thomas Röfer, Tim Laue, Jesse Richter-Klug, and Felix Thielke. B-Human team description for RoboCup 2016. In *RoboCup 2016: Robot Soccer World Cup XX Preproceedings*, Leipzig, Germany, 2016. RoboCup Federation.

[79] Niklas Rughöft. Maximum-likelihood-methode zur farbklassifikation der FUmanoid-roboter. Bachelor thesis, Freie Universität Berlin, January 2015.

[80] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3D: Learning 3D scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2009.

[81] Daniel Seifert, Lutz Freitag, Jan Draegert, Simon Gene Gottlieb, Roman Schulte-Sasse, Gregor Barth, Malte Detlefsen, Jan Bernoth, Niklas Rughöft, Michael Pluhatsch, Martin Wichner, and Raúl Rojas. Berlin United - FUmanoids team description paper for RoboCup 2015. 2015.

[82] Yaser Ajmal Sheikh, Erum Arif Khan, and Takeo Kanade. Mode-seeking by medoidshifts. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[83] Jianbing Shen, Yunfan Du, and Xuelong Li. Interactive segmentation using constrained Laplacian optimization. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(7):1088–1100, 2014.

[84] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[85] Guang Shu, Afshin Dehghan, and Mubarak Shah. Improving an object detector and extracting regions using superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3721–3727, 2013.

[86] Erik Sporns. New eyes for grace. Bachelor thesis, Freie Universitt Berlin, Juni 2016.

[87] David Stutz. Superpixel segmentation using depth information. Bachelor thesis, RWTH Aachen University, Aachen, Germany, September 2014.

[88] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 2017.

[89] Joseph Tighe and Svetlana Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *Computer Vision–ECCV 2010*, pages 352–365, 2010.

[90] Antonio Torralba. Contextual priming for object detection. *International Journal of Computer Vision*, 53(2):169–191, 2003.

[91] Leslie G Ungerleider and James V Haxby. 'what'and 'where' in the human brain. *Current opinion in neurobiology*, 4(2):157–165, 1994.

[92] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1879–1886. IEEE, 2011.

[93] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. SEEDS: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, pages 13–26. Springer, 2012.

[94] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. *Computer vision–ECCV 2008*, pages 705–718, 2008.

[95] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. *Computer Vision–ECCV 2010*, pages 211–224, 2010.

[96] Luc Vincent. Morphological grayscale reconstruction in image analysis: Applications and efficient algorithms. *IEEE transactions on image processing*, 2(2):176–201, 1993.

[97] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, 13(6):583–598, 1991.

[98] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001.

[99] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1377–1384, 2013.

[100] Hermann Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867.

[101] Shu Wang, Huchuan Lu, Fan Yang, and Ming-Hsuan Yang. Superpixel tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1323–1330. IEEE, 2011.

[102] Wenguan Wang and Jianbing Shen. Higher-order image co-segmentation. *IEEE Transactions on Multimedia*, 18(6):1011–1021, 2016.

[103] Wenguan Wang, Jianbing Shen, and Fatih Porikli. Saliency-aware geodesic video object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3395–3402, 2015.

[104] M. Wertheimer. *Laws of organization in perceptual forms.* Harcourt, Brace & Jovanovitch, London, 1938.

[105] Zhenyu Wu and Richard Leahy. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 15(11):1101–1113, 1993.

[106] Linli Xu, Wenye Li, and Dale Schuurmans. Fast normalized cut with linear constraints. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2866–2873. IEEE, 2009.

[107] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer, 2014.

[108] Kota Yamaguchi, M Hadi Kiapour, Luis E Ortiz, and Tamara L Berg. Parsing clothing in fashion photographs. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3570–3577. IEEE, 2012.

[109] Junjie Yan, Yinan Yu, Xiangyu Zhu, Zhen Lei, and Stan Z. Li. Object detection by labeling superpixels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5107–5116, 2015.

[110] Allen Y. Yang, John Wright, Yi Ma, and S. Shankar Sastry. Unsupervised segmentation of natural images via lossy data compression. *Computer Vision and Image Understanding*, 110(2):212–225, 2008.

[111] Fan Yang, Huchuan Lu, and Ming-Hsuan Yang. Robust superpixel tracking. *IEEE Transactions on Image Processing*, 23(4):1639–1651, 2014.

[112] Jian Yao, Marko Boben, Sanja Fidler, and Raquel Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2947–2955, 2015.

[113] Dingwen Zhang, Junwei Han, Chao Li, Jingdong Wang, and Xuelong Li. Detection of co-salient objects by looking deep and wide. *International Journal of Computer Vision*, 120(2):215–232, 2016.

[114] Jinxia Zhang, Krista A Ehinger, Haikun Wei, Kanjian Zhang, and Jingyu Yang. A novel graph-based optimization framework for salient object detection. *Pattern Recognition*, 64:39–50, 2017.

[115] Yuhang Zhang, Richard Hartley, John Mashford, and Stewart Burn. Superpixels via pseudo-boolean optimization. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1387–1394. IEEE, 2011.