University of Ljubljana
Faculty of Computer and Information Science

Aleksandar Dimitriev

A Markov random field based autonomous image segmentation

BACHELOR'S THESIS

UNDERGRADUATE UNIVERSITY STUDY PROGRAMME
COMPUTER AND INFORMATION SCIENCE

Mentor: doc. dr. Matej Kristan

Ljubljana 2014

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Aleksandar Dimitriev

Avtonomna segmentacija slik z Markovim slučajnim poljem

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Matej Kristan

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

The Faculty of Computer and Information Science issues the following thesis:

Analyze the problem of autonomous image segmentation designed for cases in which the object of interest occupies a significant portion of the image. Formulate the segmentation as inference in Markov random fields and apply discriminative approaches for modelling object visual properties. In addition, propose appropriate visual features for efficient representation of the visual properties required for segmentation. Analyze the proposed approach on a standard publicly-available dataset and evaluate it against related approaches.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Analizirajte problem avtonomne segmentacije slik, kjer objekti zanimanja predstavljajo dovolj velik del slike. Za segmentacijski model uporabite Markovo slučajno polje in ga povežite z diskriminativnimi postopki modeliranja izgleda objektov. Predlagajte tudi izbiro primernih značilnic za opis vizualne informacije za segmentacijo. Predlagani postopek segmentacije analizirajte na standardnih javno dostopnih zbirkah in ga primerjajte s sorodnimi deli.

Rad bi se zahvalil svojemu mentorju, doc. dr. Mateju Kristanu za njegov čas, ideje in potrpežljivost pri izdelavi diplomske naloge.

Za nenehno podporo in spodbudo tekom študija, se zahvaljujem tudi moji mati Lenki, in moji punci Ivani.

I would like to thank my mentor, doc. dr. Matej Kristan, for his time, ideas and patience during the writing of the thesis.

For continuous support and encouragement during my studies I thank both my mother Lenka, and my girlfriend, Ivana.

Contents

Table of acronyms

acronym	angleško	slovensko
EM	expectation maximization	Maksimizacija pričakovanja
GMM	Gaussian mixture model	Gaussova mešanica
MRF	Markov random field	slučajno polje Markova
CRF	conditional random field	pogojno slučajno polje Markova
HMM	hidden Markov Model	prikriti model Markova
HMRF	hidden Markov random field	prikrito slučajno polje Markova
SVM	support vector machine	metoda podpornih vektorjev
SLIC	simple iterative linear clustering	enostavno linearno ponavljajoče grozdenje
COLOR CHILD	color moments augmented cumulative histogram-based image local descriptor	lokalni slikovni deskriptor na osnovi kumulativnega histograma razširjen z barvnimi momenti

Abstract

Image segmentation is a widely-researched topic with many algorithms available. Our goal is to segment an image, in an unsupervised way, into several coherent parts with the help of superpixels. To achieve that, we propose an iterative segmentation algorithm. The algorithm models the image by a Markov random field, whose nodes are the superpixels, and each node has both color and texture features. The superpixels are assigned labels according to their features with the help of support vector machines and the aforementioned MRF. As a first step, the algorithm oversegments an image into hundreds of superpixels and extracts features. This is followed by expectation maximization iterations, in which both the labels and the classifier parameters are estimated. In each iteration the SVMs are trained and produce outputs, that are combined with spatial information encoded in the MRF. Each iteration removes labels with a low number of superpixels, and similar labels are merged. The tentative segmentations after each iteration are refined and the result is a segmentation of an image into several semantically meaningful regions with requiring any user input. The segmentation algorithm was tested on a standard evaluation database, and performs on par with state-of-the-art segmentation algorithms in F-measures. In terms of oversegmentation, our approach significantly outperforms the state of the art by greatly reducing the oversegmentation of the object of interest.

Keywords: segmentation, support vector machines, SVM, Markov random field, MRF, unsupervised learning.

Povzetek

Segmentacija slik je zelo raziskovano področje, za katero so na voljo številni algoritmi. Naš cilj je segmentacija slike s pomočjo superpikslov na več skladnih delov in na nenadzorovan način. Da bi to dosegli, predlagamo iterativni segmentacijski algoritem. Algoritem predstavlja sliko kot slučajno polje Markova (MRF), katerega vozlišča so superpiksli, ki imajo barvne in teksturne atribute. Superpikslom dodelimo oznake na podlagi njihovih atributov s pomočjo metode podpornih vektorjev (SVM) in že omenjenega MRF. Kot prvi korak algoritem razdeli sliko na več kot sto superpikslov in izračuna barvne in teksturne atribute. Temu sledijo iteracije maksimizacije pričakovanja, v katerih se hkrati ocenjuje oznake in parametre klasifikatorja. V vsaki iteraciji SVM so usposobljeni in ustvarjajo rezultate, ki jih algoritem združuje s prostorskimi informacijami, zakodiranimi v MRF. Vsaka iteracija odstrani oznake, ki imajo premalo superpikslov, in združi podobne oznake. Negotovo segmentacijo po vsaki iteraciji se izboljšuje in rezultat je segmentacija slike na več semantično smiselnih delov, brez pomoči uporabnika. Algoritem je bil testiran na segmentacijsko podatkovno bazo in F ocene so podobne najsodobnejšim algoritmom. Glede fragmentacije slike naš pristop bistveno prekosi stanje tehnike z zmanjšanjem števila segmentov, iz katerih je sestavljen predmet zanimanja.

Ključne besede: segmentacija, metoda podpornih vektorjev, SVM, slučajno polje Markova, MRF, nenadzorovano učenje.

Razširjeni povzetek

Segmentacija je razdelitev slike na majhno število delov, kjer so piksli, iz katerih je sestavljen ta del, medsebojno podobni, medtem ko se bistveno razlikujejo od pikslov drugih delov slike. Je zelo raziskovano področje, za katero so na voljo številni algoritmi, ker je njihova praktična uporaba velika, na primer za področje strojnega vida, detekcije objektov, medicinsko upodabljanje slike itd. Kljub temu, da je bilo veliko segmentacijskih algoritmov razvitih v zadnjih dveh desetletjih, so še vedno vir številnih člankov, ker računalnikom segmentacija ni tako enostavna kot je ljudem.

Ko človek vidi sliko, jo avtomatsko razdeli na več semantičnih delov, recimo ozadje in ospredje, ki sta oba lahko razdeljena na več objektov, kot je nebo, tla itd. Človeku pomaga veliko različnih napotkih, kot avtomatska ocena globine in detekcija objektov, medtem ko računalnik vidi samo dvodimenzionalno matriko intenzitet oziroma trodimenzionalno, če upošteva barve.

Zaradi tega je bilo raziskovanih veliko različnih načinov segmentacije, na primer: detekcija robov oziroma mej, zaporedno združevanje podobnih regij v sliki na podlagi različnih atributov, kot je barva in tekstura, uporaba pikslov kot vozlišč grafa in naknadni rez tega grafa ter razvrščanje pikslov v več skupin na podlagi že omenjenih atributov.

Zelo uspešna metoda, ki se je pred kratkim pojavila, je razčlenitev slike na stotine superpikslov. To je samo ime za majhne dele slike, v katerih so si piksli medsebojno zelo podobni. Takšna segmentacija se pogosto uporablja kot prvi korak, kjer se algoritmi naknadno izvajajo na podlagi teh superpikslov namesto navadnih pikslov. Metode, ki temeljijo na grafih, so kot začetni korak uporabljale zlasti superpiksle. Naš algoritem jih tudi uporablja na enak način, s katerim zgradimo neusmerjeni graf, ki mu pravimo pogojno naključno polje Markova.

Naš cilj je bil razviti splošni nenadzorovani segmentacijski algoritem, ki lahko razčleni katero koli sliko na več skladnih delov. Nenadzorovani način pomeni, da uporabnik ne pove, koliko segmentov naj ima slika ali kje se približno nahajajo. Da bi to naredili, smo formulirali iterativno metodo, ki uporablja metode podpornih vektorjev in pogojno slučajno polje Markova. Postopek je naslednji: najprej se naredi segmentacijo slike s pomočjo SLIC superpikslov na približno 300 superpikslov. SLIC [1] je izboljšava razvrščanja z voditelji in ima manjšo oziroma linearno, časovno kompleksnost ter poda boljše rezultate zaradi upoštevanja prostorske informacije.

Sledi izračun barvnih in teksturnih atributov vsakega superpiksla s pomočjo COLOR CHILD deskriptorja. COLOR CHILD [4] je najsodobnejši algoritem, ki je sestavljen iz dveh delov: barvne in teskturne informacije. Barvni atributi so povprečje, standardni odklon in tretji centralni moment vseh treh barvnih kanalov iz izbranega prostora. Drugi del deksriptorja je sestavljen iz dveh teksturnih komponent. Grobo rečeno, prva komponenta je izračun odvoda slike v x in y smer, ki je velik v teksturnih regijah in robovih. Druga komponenta teksturnega dela je orientacija odvoda, ki nam tudi da informacije o tesksturi superpiksla. Oba teksturna dela sta kvantizirana, ker se na koncu dobi dvo-dimenzionalni histogram, ki ju združi. COLOR CHILD je sestavljen iz tega histograma in že omenjenih barvnih momentov.

Sledi glavni del algoritma, ki je iterativen in sestavljen iz dveh korakov: ocena verjetnosti, da vsak superpiksel pripada vsakemu segmentu, in posodobitev parametrov na podlagi zgoraj navedene ocene, analogno algoritmu maksimizacija pričakovanja (EM) [14]. Oceno verjetnosti določimo s pomočjo metode podpornih vektorjev (SVM) [12] in naključnega polja Markova (MRF). Ker je možnih segmentov več, SVM uporablja "eden proti vsem" strategijo, ki sestoji iz grajenja N klasifikatorjev, kjer je N število oznak, oziroma segmentov (na začetku je enako številu superpikslov). SVM klasificira superpiksle na podlagi barvnih in teksturnih atributov. Vsaka iteracija odstrani oznake, ki imajo premalo superpikslov, in proste superpiksle SVM razvrsti med ostalimi.

Glede upoštevanja strukturnih informacij, kot je bližina superpikslov v sliki, uporabljamo naključno polje Markova (MRF), ki je zgrajeno na njih. MRF je neusmerjen graf, v katerem imajo vozlišča, ki so superpiksli, povezavo s tistimi superpiksli, ki so sosedje, kjer so sosedje superpiksla vsi, ki se ga dotikajo. MRF kaznuje tiste sosede, ki imajo različne oznake, sorazmerno z energijo te povezave, kjer je energija obratna barvni podobnosti teh dveh sosedov.

Bolj formalno; pri vsaki iteraciji zgradimo SVM klasifikator za vsako oznako, pri katerih so pozitivni primeri superpiksli s to oznako, negativni primeri pa so vsi ostali. Izračunamo verjetnost pripadnosti vsakega superpiksla vsaki oznaki. Dobljene verjetnosti spreminjamo z MRF. Vplivajo sosedje, ki so bolj barvno podobni oznaki, ki jo imajo. Na koncu vzamemo za vsak superpiksel oznako z največjo verjetnostjo in dobimo trenutno segmentacijo. Ker se oznake lahko spreminjajo pri vsaki iteraciji, imamo zdaj nove parametre, s katerimi spet zgradimo nove SVM klasifikatorje v naslednji iteraciji.

```
Izvedi SLIC in razdeli sliko na superpiksle;
for vsak superpiksel do
    Izračunaj barvne in teksturne atribute;
end
while ni konvergiral do
    for vsako oznako do
        Izračunaj verjetnost pripadnosti vsakega superpiksla tej oznaki;
        Združi dobljeno verjetnost z verjetnostjo predhodne iteracije;
        Posodobi združeno verjetnost z MRF in pridobi končne
        verjetnosti;
    end
    Vsakemu superpikslu dodeli oznako z največjo verjetnostjo;
    for vsako oznako do
        Odstrani, če nima superpikslov;
        Zgradi SVM klasifikator za njo;
    end
end
```

Algorithm 1: Algoritem za segmentacijo slike

Poleg tega uporabljamo predhodno verjetnost pri izračunu končne verjetnosti, ki je združena končna verjetnost vseh prejšnjih iteracij. Na začetku je verjetnost vsakega superpiksla za vse oznake enaka. Rečemo lahko tudi, da je to avtoregresijski model. Povzetek algoritma je podan na Sliki 1.

Na koncu je bil algoritem testiran na segmentacijsko podatkovno bazo, ki je sestavljena iz slik, na katerih je en predmet in ozadje. Rezultati so podobni najsodobnejšim algoritmom, segmentacije, ki jih dobimo, pa so podobne človekovi. Natančnost je blizu rezultatov ostalih algoritmov, vendar je povprečno število segmentov, iz katerih je sestavljen predmet oziroma ospredje, bistveno manjše in je blizu 1, posledično pa je fragmentacija najmanša. Torej naš algoritem uspešno segmentira sliko na več delov, ki se medsebojno razlikujejo, medtem ko so (super)piksli znotraj enega segmenta podobni.

Chapter 1

Introduction

1.1 Motivation

The problem of image segmentation is well-established in the field of computer vision. It entails partitioning an image into multiple fragments, the number of which can vary from two to several hundred. When the number is on the low side, e.g. fewer than ten segments, each segment can be said to contain a meaningful part of the image, e.g. sky, ground or tree. On the other hand, when the number of segments is on the order of a hundred or more, the image is said to be partitioned into superpixels, which are groups of pixels that are similar in color, texture, or some other attribute.

The motivation behind this work was to create a general-purpose algorithm that works on any image and provides a segmentation similar to what a human would present. Such an algorithm would be of use to many fields where it can be integrated with domain knowledge, e.g medical imaging, object detection, content-based image retrieval etc.

1.2 Our approach and contributions

In this thesis, we use both kinds of segmentation, first clustering the hundreds of thousands of pixels into a few hundred superpixels, then iteratively applying another algorithm to merge those superpixels into a few meaningful image segments. To achieve this goal, we concentrated our work on the latter step of the segmentation and used an existing algorithm, namely Simple Linear Iterative Clustering [1], or SLIC, to segment the image into several hundred segments - also called superpixels. Color and texture features for each one are computed using a state-of-the-art local descriptor called COLOR CHILD [4] (COLOR moments augmented Cumulative Histogram-based Image Local Descriptor), and iteratively merged until the visual similarity between the segments is sufficiently high. The superpixels are merged according to the output of a classifier, namely a support vector machine (SVM) [12], combined with a Markov random field (MRF) to encode structural properties and enforce local regularization in the segmentation. Our method is tested on a standard dataset and achieves state-of-the-art results.

1.3 Related work

The topic of image segmentation has generated many useful results and still remains a hot topic. Relatively recently, however, the use of superpixels as a way to (over)segment an image in a first step has emerged in many segmentation approaches [45, 28, 6, 25]. Although segmenting the image into a few regions is still the main point, there has been a lot of work in superpixel segmentation specifically, beginning with normalized cuts [39] and followed by [16], which are graph-based algorithms. There are also mode-seeking algorithms, namely mean shift [11], quick shift [43], and SLIC [1]. All of the aforementioned algorithms have been utilized for superpixel segmentation, but there have been significant results without involving superpixels, e.g. GrabCut [35] and others [26, 33, 5, 22].

Aggregating small regions of the image into larger segments has also been proven to yield good segmentations [3], using shape and texture cues [38], or by using contours [6]. Applying a superpixel segmentation as a first step is very useful for graph-based algorithms like [46], since it offers a considerable speed-up when superpixels are used as nodes in the graph instead of superpixels. Many have used this as a first step, e.g. in tree graph partitioning [45], segmentation by data compression [49] [28], the aforementioned contour algorithm [6], a greedy merge algorithm [34], and a bipartite graph partioning approach [25] which uses two types of superpixels, [11] and [16], as the first step.

The most related work is probably [20], even though its goal is object detection, since it uses CRFs to encode structural information, an SVM classifier, as well a preliminary superpixel segmentation as the first step. Another very related work is a segmentation algorithm [42], which uses a GMM with a HMRF, first presented in [51], for structural properties, but does not use superpixels.

1.4 Structure

The rest of this thesis is structed as follows: Chapter 2 describes the theory behind our algorithm. It includes the SLIC superpixel segmentation algorithm, the COLOR CHILD descriptor that computes color and texture features, Markov random fields, their extensions and the expectation maximization algorithm for inference used on those models, and ends with a brief overview of support vector machines. It is followed by our segmentation algorithm in Chapter 3, which incorporates all of the previous methods. Next are some experimental results on a segmentation database in Chapter 4, followed by the conclusion and future work in Chapter 5.

Chapter 2

Methods used

2.1 SLIC superpixels

Simple Iterative Linear Clustering, henceforth SLIC, is a state-of-the-art algorithm that segments an image into a desired number of groups of pixels called superpixels. The usefulness of a superpixel is its ability to remove redundancy in color or texture information in neighboring pixels by grouping them together. Consequently, it reduces the computational cost of any algorithm that works on the pixel level, especially graph-based methods.

For example, if every pixel in a 300-by-400-pixel image is a node, then the resulting graph has $N = 120000$ nodes. Grouping similar pixels, however, can yield several hundred superpixels, if we set the number of desired superpixels to 300. This graph, on the other hand, has only $N = 300$ nodes, which can dramatically reduce the computational cost of even a linear algorithm that works on graph nodes or edges.

Although SLIC is an adaptation of k-means, it has reduced complexity and yields better results due to two specific improvements. The superpixels should be compact, and computing the distances between estimated centers for all pixels in the image is undesired. Instead, they are computed only for a region proportional to the size of the superpixel. Secondly, Euclidean distance in the five dimensional feature space of color and space, or x and y coordinates, is a poor measurement because it does not offer a way to control the compactness of a superpixel. Therefore, SLIC uses a weighted distance measure that combines a Euclidean distance in the three-dimensional colorspace YCbCr and the two-dimensional cooordinate space, computed as follows:

$$d_{YCbCr} = \sqrt{(Y_i - Y_j)^2 + (Cb_i - Cb_j)^2 + (Cr_i - Cr_j)^2} \tag{2.1}$$

$$d_{xy} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2.2}$$

$$D = d_{YCbCr} + \frac{m}{S} d_{xy} \tag{2.3}$$

where Y_i,Cb_i and Cr_i denote the Y,Cb and Cr channels of the i^{th} pixel. The parameter m controls the compactness of the superpixel. Higher values give more weight to the Euclidean coordinate distance and produce uniformly shaped superpixels, and is $m = 10$ is set by the authors in the original paper [1]. Lastly, $S = \sqrt{\frac{N}{K}}$, where N is the number of pixels and K is the number of desired superpixels, regardless of the amount of texture in the image or its size.

The value of S is also used to limit the search space. Since we expect each superpixel to be roughly of size S-by-S, we only compute the distance measure for each pixel in the neighborhood of size $2S$-by-$2S$, which greatly reduces the complexity. When initializing the cluster centers, the coordinates are perturbed in a $3 - by - 3$ neighborhood to the lowest gradient, which is the combined gradient in the x and y direction, computed as follows:

$$G(x, y) = ||I(x + 1, y) - I(x - 1, y)||^2 + ||I(x, y + 1) - I(x, y - 1)||^2 \tag{2.4}$$

This is done to prevent edge pixels from becoming cluster centers. The pseudocode for the SLIC algorithm is summarized in Algorithm 2.

While regular k-means has time complexity on the order of $O(N * K * I)$, where N is the number of pixels, K is the desired number of superpixels and I is the number of iterations, SLIC has $O(N)$ time complexity. This is because the number of iterations is fixed to 10 by the authors, and K is a varying number less than 8, because each pixel is at most in the neighborhood of 8 cluster centers.

In this thesis we use SLIC as a preliminary segmentation step, so that a classifier like SVM is trained for each superpixel instead of each pixel, but more importantly,

```
Initialize cluster centers $C_k = [Y_k \, Cr_k \, Cb_k \, x_k \, y_k]^T$ in a grid S pixels wide;
Perturb each cluster center to the lowest gradient;
while not converged do
    for each cluster center $C_k$ do
        Compute the distance shown in Eq. (2.3) to each pixel in its
        $2S - by - 2S$ neighborhood;
    end
    Compute new cluster centers;
end
```

Algorithm 2: Simple Linear Iterative Clustering (SLIC) as proposed in [1]

the MRF used has a small number of nodes (the number of superpixels, instead of pixels) that is fixed to 300 for an image of any size. As can be seen, this number is small enough to ensure computational efficiency, yet large enough to properly describe the structure of the image. Three different values of the desired number of superpixels, more specifically 100, 300 and 900, can be seen in Fig. 2.1, respectively.

2.2 COLOR CHILD

Color moments augmented Cumulative Histogram-based Image Local Descriptor, or COLOR CHILD [4], is a local image descriptor that uses both color and texture. Descriptors are useful for many computer vision tasks like object recognition and tracking, and play a particularly important role in image segmentation. In our approach, the COLOR CHILD descriptor is used to encode the color and texture properties of image regions. We provide an overview of the descriptor in this section and refer the reader to [4] for further details.

2.2.1 Color features

The descriptor is comprised of two distinct types of features, color and texture. The color features come from the first, second, and third moments of the image colorspace (in our case, YCbCr), or the mean, standard deviation, and skewness.

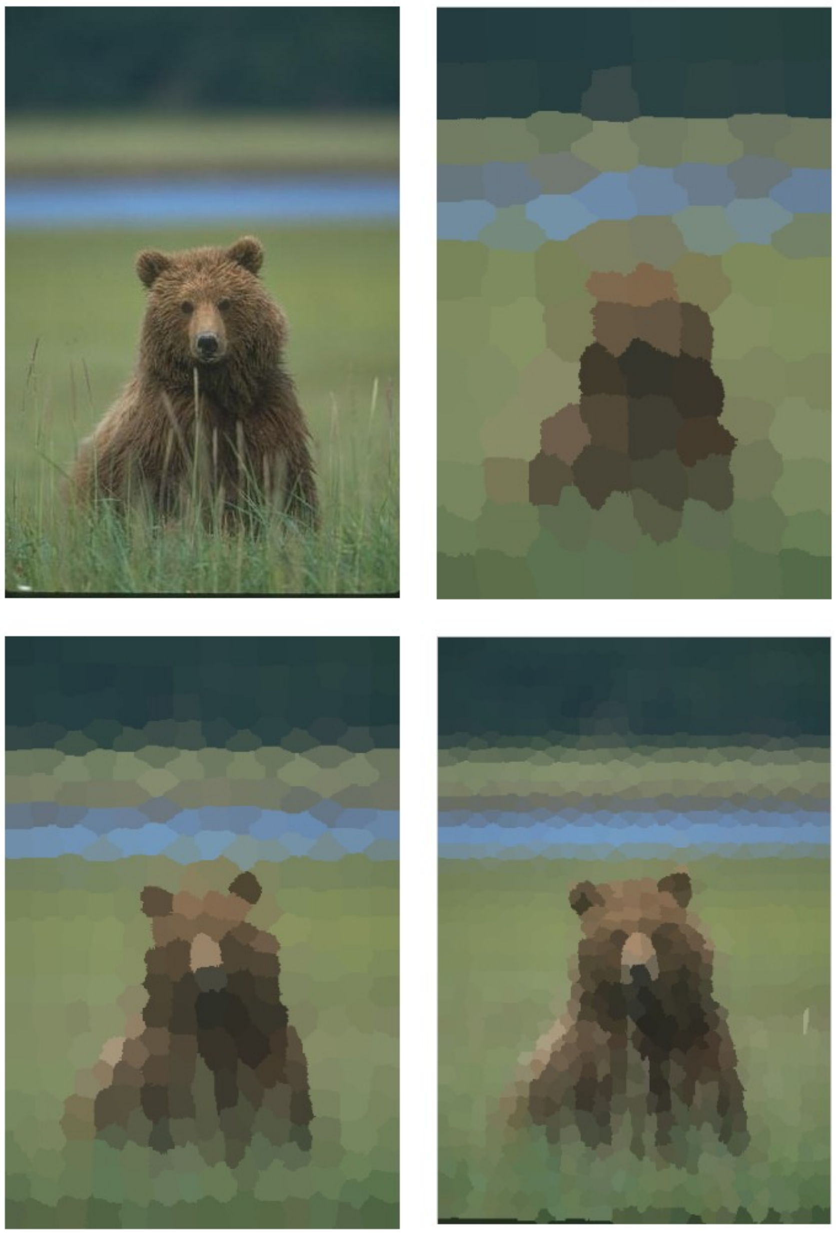

Figure 2.1: Clockwise from top left: Original image from [27], SLIC with 100, 300, and 900 superpixels.

These are calculated for each color channel as follows:

$$\mu_i = \frac{1}{N}\sum_{j=1}^{n} p_{ij}, \tag{2.5}$$

$$\sigma_i = \sqrt{\frac{1}{N}\sum_{j=1}^{n}(p_{ij} - \mu_i)^2} \tag{2.6}$$

$$s_i = \sqrt[3]{\frac{1}{N}\sum_{j=1}^{n}(p_{ij} - \mu_i)^3} \tag{2.7}$$

where p_{ij} is the value of the i^{th} color channel at the j^{th} pixel, and N is the number of image pixels. Thus, we obtain nine color features for each pixel.

2.2.2 Texture features

An equally important part of this descriptor are the texture features, which are computed as a combination of two components, namely, differential excitation and gradient orientation. The resulting features are a two-dimensional histogram that quantizes and combines the two components, which are explained in more detail in the following sections.

Differential excitation

The Laplacian of Gaussian is a well-known blob detection technique in the field of computer vision [18]. It consists of smoothing (or blurring) the image with a Gaussian kernel (Eq 2.8) and applying the Laplace operator (Eq 2.10), which is the sum of the second partial derivatives, at each pixel of the blurred image. If the resulting image is normalized with the pixel intensity (or value) and insert it into the arctan function to reduce noise (Eq 2.11), then the differential excitation component can be obtained.

$$h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.8}$$

$$g(x,y) = h(x,y) \otimes f(x,y) \tag{2.9}$$

$$\nabla^2 g = \frac{1}{\pi\sigma^4}(\frac{x^2+y^2}{2\sigma^2} - 1)e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.10}$$

$$\xi(x,y) = arctan(\frac{\nabla^2 g}{f(x,y)}) \tag{2.11}$$

where (x,y) are the image coordinates, $f(x,y)$ the intensity at that position, $h(x,y)$ the two-dimensional Gaussian (smoothing) function, ∇^2 the Laplacian, or second derivative operator, and $\otimes$ the convolution operator. Then the aforementioned equations present the step-by-step algorithm for producing the differental excitation, which can be seen in Fig. 2.2.

The differential excitation measures the "roughness" of an image, yielding high values in textured regions. The intuition behind it is simple. Computing the first derivative of an image produces high values where pixels suddenly change intensity in any direction and is commonly used for edge detection. Therefore, the second derivative, or the Laplacian, tells us where in the image we have changes in edge detection, which is usually in highly textured regions. In those regions the pixel intensity in any direction vary wildly, and this component registers that. The image is quantized into M bins, so that a histogram can be be built in the last step.

Fractional gradient orientation

Although discriminating between flat and textured regions provides useful information, the orientation of the gradient image (the direction in which the intensity of a pixel changes) also plays an important part. COLOR CHILD employs a fractional derivative, i.e. gradient, that is computed using the following equations. Using fractional calculus, for a general function $f(x)$ and $0 < \alpha < 1$, the complete fractional derivative is:

$$\frac{d^\alpha}{dx^\alpha} f(x) = D^\alpha f(x) = \frac{1}{\Gamma(1-\alpha)} \frac{d}{dx} \int_0^x \frac{f(t)}{(x-t)^\alpha} dt \tag{2.12}$$

where D is the derivation operator, α is the desired fractional derivative, and Γ is the extension of the factorial function. Now if $f(x) = y$, any fractional derivative of

$\frac{(\alpha^2-\alpha)}{2}$	$-\alpha$	0	α	$\frac{(\alpha^2-\alpha)}{2}$
$(\alpha^2-\alpha)$	-2α	0	2α	$(\alpha-\alpha^2)$
$\frac{3(\alpha^2-\alpha)}{2}$	-3α	0	3α	$\frac{3(\alpha-\alpha^2)}{2}$
$(\alpha^2-\alpha)$	-2α	0	2α	$(\alpha-\alpha^2)$
$\frac{(\alpha^2-\alpha)}{2}$	$-\alpha$	0	α	$\frac{(\alpha-\alpha^2)}{2}$

Table 2.1: Fractional differential mask of size $5x5$ for x direction

a function can be restated with the Fractional differential finite impulse (FIR) filter transfer function as follows:

$$D^{\alpha}(y) = (\frac{1-y^{-1}}{T})^{\alpha} \tag{2.13}$$

where T is the sampling period. Using the binomial series expansion $(1+x)^n = 1 + nx + \sum_{k=2}^{\infty} \frac{n!}{k!(n-k)!}x^k$, we can rewrite Eq 2.13 as:

$$D^{\alpha}(y) = \frac{1}{T^{\alpha}}(\sum_{k=0}^{\infty}(-1)^k \frac{\Gamma(\alpha+1)}{\Gamma(k+1)\Gamma(\alpha-k+1)}y^{-k} \tag{2.14}$$

For any real-world application, the sum is truncated to a predetermined number N, and the derivation is discretized for application in the two-dimensional and discrete image domain. Yang et al. [50] provide the fractional derivative masks of size $5x5$ for x and y direction, shown in Table 2.1 and 2.2, respectively.

If the outputs of convolving the aforementioned masks in x and y direction are ν_x and ν_y, then the fractional gradient orientation for a pixel at coordinates (x, y) is obtained by taking the inverse tangent function of their ratio, or:

$$\theta(x, y) = arctan(\nu_y, \nu_x) \tag{2.15}$$

$\frac{(\alpha^2-\alpha)}{2}$	$(\alpha^2-\alpha)$	$\frac{3(\alpha^2-\alpha)}{2}$	$(\alpha^2-\alpha)$	$\frac{(\alpha^2-\alpha)}{2}$
$-\alpha$	-2α	-3α	-2α	$-\alpha$
0	0	0	0	0
α	2α	3α	2α	α
$\frac{(\alpha-\alpha^2)}{2}$	$(\alpha-\alpha^2)$	$\frac{3(\alpha-\alpha^2)}{2}$	$(\alpha-\alpha^2)$	$\frac{(\alpha-\alpha^2)}{2}$

Table 2.2: Fractional differential mask of size $5x5$ for y direction

The angle, or orientation, is quantized into T dominant directions, so that a histogram can be properly built. The value of the fractional derivative α is set to 0.6 in this thesis. The resulting gradient orientation image after quantization can be seen in Fig. 2.2.

Texture descriptor generation

The final step in generating the texture features is to create a one-dimensional histogram. Since each image pixel can have T orientations and M excitations, the resulting histogram image is a two-dimensional $T - by - M$ histogram. By simply concatenating each dimension, a one-dimensional texture histogram is obtained that, together with the color moments, presents the output of this descriptor.

2.3 Markov random fields

Widely used in low-level vision problems, Markov random fields, or MRFs, provide a useful way of encoding spatial dependencies of image locations. They have been extensively utilized in problems like image restoration [17, 7] and completion [36], texture synthesis [29, 15], structure from motion [21], stereo vision [41, 37], and of course, segmentation [40, 51, 48, 20]. A brief introduction and background follows,

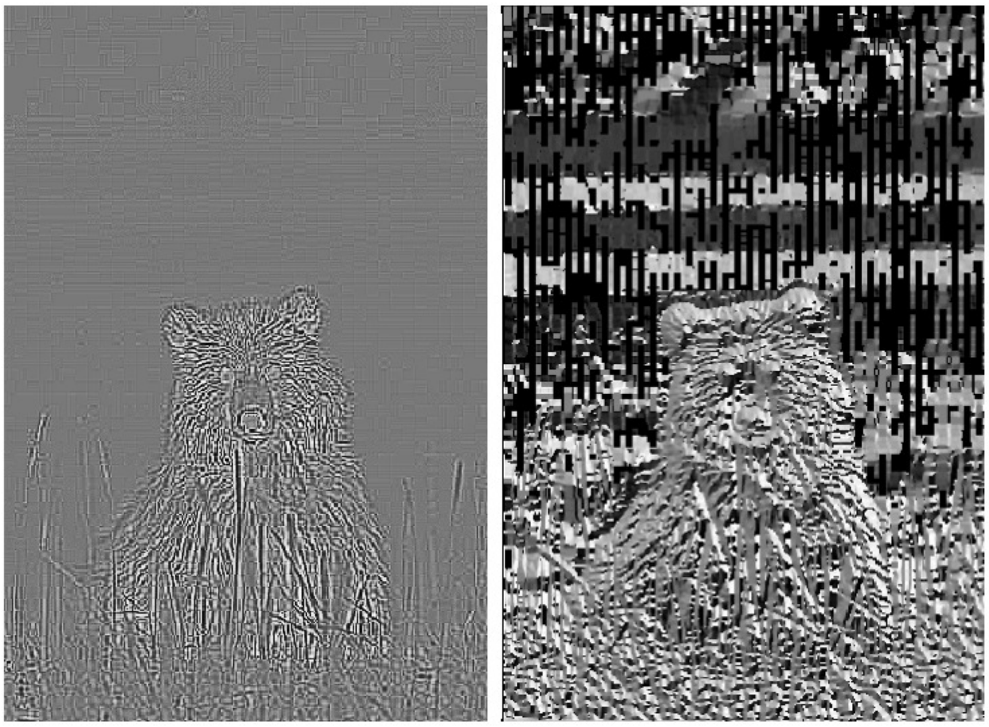

Figure 2.2: The texture components of COLOR CHILD, namely, the differential excitation, shown on the left, and the fractional gradient orientation with $\alpha = 0.6$, shown on the right.

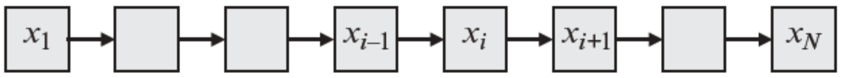

Figure 2.3: A graphical representation of a Markov chain.

though the reader may consult [24] for more information.

2.3.1 Background

The Markov property of a temporal model means that future states depend only on the current state, i.e. a future state depends only on the present and not the past states. There are variations where a state depends on the past m states, i.e. the Markov property is of order m, or even variable-order Markov models, though the most widely used are Markov models that depend only on the previous step.

We can also replace the temporal dependency by a spatial dependency, e.g. in a graph. The Markov property here means that a node is conditionally independent of all other nodes in the graph, given its neighbors (the nodes with which it shares an edge).

2.3.2 An example of a temporal Markov chain

To better illustrate the aforementioned Markov property, we shall consider the simplest Markov model, namely, the Markov chain of order one, seen in Fig. 2.3. It is a system where the the transition from each state depends only on the previous one. It should not, however, be confused with a finite-state machine, or FSM. The differences are that the state space of a Markov model can be infinite as opposed to FSMs, and the state transitions of a Markov chain are probabilistic. Both of them are generalized by probabilistic automata [31].

More formally, the Markov property of the chain is defined as follows:

$$P(X_n = x_n | X_1 = x_1, X_2 = x_2, ..., X_{n-1} = x_{n-1}) = P(X_n = x_n | X_{n-1} = x_{n-1}) \tag{2.16}$$

where the probability of any combination of states (in the state space) is always positive, or $P(X_1 = x_1, ..., X_n = x_n) > 0$.

Example

A textbook example [8] is a simplified weather model, with the only possible states, or days, being $X_i \in \{sunny, rainy\}$. Of course, the weather can depend on many previous days and have other states, but for illustrative purposes we shall consider only those two, and a dependence of order one, i.e. only on the previous day. The set of conditional probabilities is the following transition matrix:

	sunny	*rainy*
sunny	0.7	0.3
rainy	0.4	0.6

(2.17)

where the probability of a transition from state i to state j is given by the value in position a_{ij} in the matrix. Predicting the next state in the chain is trivial, as it only requires selecting the highest probability in the row whose state equals the previous. Although there is only a first-order dependency defined explicitly, there are long-range dependencies between states. For example, to get the most probable state two days from now, we must multiply the probabilities as follows:

$$P(X_{i+2} = x_{i+2}|X_i = x_i) = P(X_{i+2} = x_{i+2}|X_{i+1} = x_{i+1}) * P(X_{i+1} = x_{i+1}|X_i = x_i) \tag{2.18}$$

which shows an implicit long-range dependency. The most probable state can also be obtained by multiplying the transition matrix with itself and again selecting the highest probability in the row corresponding to the the first state.

In this simple case we could directly observe the weather and easily predict the next state. The question that arises, however, is what to do when the desired prediction states are unobservable, or hidden, and only some other output is directly visible?

2.3.3 Hidden Markov Model

A Hidden Markov Model can be thought of as a simple dynamic Bayesian network. It was popularized by Rabiner [32] when inference algorithms were invented. The essence of a Hidden Markov Model (HMM) is that we observe values that are not perfectly corelated with the data we are trying to perceive, but are nonetheless useful in helping us determine the underlying hidden observations. They are widely used, since many processes are not fully observable. Visually, a Hidden Markov Model can be seen in Fig. 2.4, where $z = (z_1, z_2, ..., z_n)$ denote uncertain observations or measurements, and $x = (x_1, x_2, ..., x_n)$ denote the hidden values to be inferred. For example, in speech processing, where HMMs are widely used, we are trying to infer the sequence of words or phonemes, while only having access to spectral data. In this case, the possible values of x include all phonemes, and the observed and noisy audio signal constitutes z.

Formally, a Hidden Markov Model of the first order satisfies the following properties:

$$P(X = x \mid Z = z) \propto P(Z = z \mid X = x)P(X = x) \tag{2.19}$$

$$P(Z_i = z_i \mid X = x) = P(Z_i = z_i|X_i = x_i) \tag{2.20}$$

$$P(Z = z \mid X = x) = P(Z_n = z_n \mid X_n = x_n)...P(Z_1 = z_1 \mid X_1 = x_1) \tag{2.21}$$

where $P(X = x) = P(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$ is just an abbreviation of the joint probability. In the previous example of a Markov chain, these probabilities sufficed to create the model. Eq. 2.19 is obtained by applying the Bayes theorem $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$. Eq. 2.20 stems from the fact that observing z_i at state x_i is dependent only on that state and the conditional probability can be computed as shown in Eq. 2.21.

For a Markov chain, only a single transition matrix is required to describe the data. In the case of HMMs, prior probabilities are also required, since we do not know the starting state, and observation probabilities for each hidden state, or an x-by-z matrix.

To obtain the most likely sequence of hidden states, the Viterbi algorithm [44] can be applied on the aforementioned data. It is beyond the scope of this thesis,

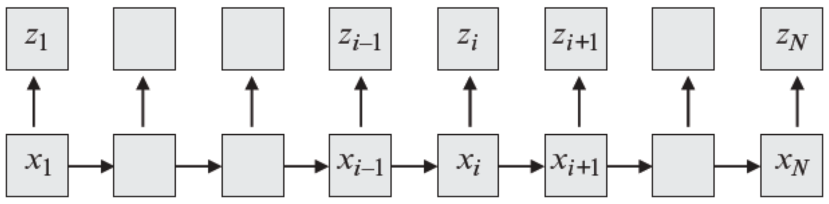

Figure 2.4: A graphical representation of a Hidden Markov Model.

however, since we are interested in a multi-dimensional and undirected HMM, which uses different algorithms. The next example serves to illustrate HMMs in contrast to Markov chains.

Example

Previously, we could directly observe the weather and easily compute the most probable next state. But often we can only observe a by-product, e.g. in the case of the weather, the actions of an individual who acts according to the hidden states, which he can observe, but we can not. In this example, the sunny and rainy states are hidden, or $X \in sunny, rainy$, and we can observe three actions, $Z \in walk, shop, clean$. To predict the next state, we must first determine the most probable sequence of states preceding it, which first requires the output probabilities:

	walk	*shop*	*clean*
sunny	0.6	0.3	0.1
rainy	0.1	0.4	0.5

(2.22)

as well as the transition matrix, previously defined in Eq. 2.17, and starting probabilities, Eq. 2.23.

	sunny	*rainy*
start	0.4	0.6

(2.23)

Alternatively, we can represent all three matrices in a single diagram, as shown in Fig. 2.5. It is relatively straightforward to now compute the most likely sequence,

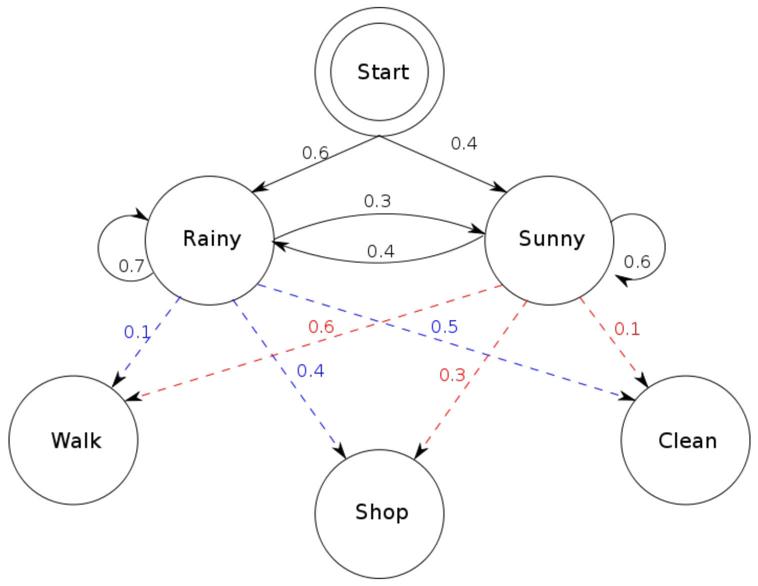

Figure 2.5: A graphical representation of the probabilities in a HMM.

though we are still using a model that has only a one-way temporal dependency of the states. The next section delves into spatial dependencies.

2.3.4 Markov random fields

In image processing, encoding structural information into a model is often needed. Also, in the example of a Markov chain, there is a one-dimensional dependency in one direction, whereas an image is a two-dimensional undirected graph, where each (super)pixel is a node connected by edges with its neighbors, or adjacent pixels. Therefore, a Markov random field can be thought of as a generalization of a Markov chain in multiple dimensions whose edges are undirected.

Now that the we have spatial dependencies, we need to determine their extent. Something equivalent to the first-order Markov chain would be a four-connected or eight-connected neighborhood, where a pixel is dependent only on its immediate neighbors, or in the case of superpixels, it is dependent on those that it shares a boundary with. The three such configurations are shown, respectively, in Fig. 2.6.

Formally, an MRF consists of an undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is the

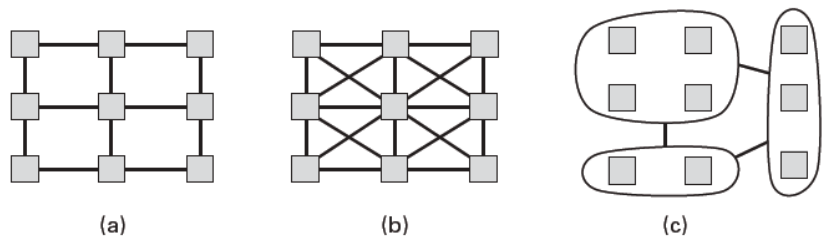

Figure 2.6: Graph of an image for MRF. a) four-connected neighborhood. b) eight-connected neighborhood. c) superpixel neighborhood consisting of bordering superpixels.

set of all nodes and $\mathcal{E}$ the set of edges. The neighbor set of any node is defined as all the nodes that share an edge with it (2.24). Given its neighbors, a node is conditionally independent of every other node in the graph. This is the Markov property for an MRF, expressed as follows:

$$N_n = \{m \in \mathcal{N} \mid (n,m) \in \mathcal{E}\}, \tag{2.24}$$

$$P(X_n = x_n \mid X = x) = P(X_n = x_n \mid X_N = x_N), \tag{2.25}$$

, however $P(X_n = x_n \mid X_N = x_N)$ is only the local distribution for each node. How do we then compute the global joint probability distrubution? According to the Hammersley-Clifford theorem [23] , it can be computed as the exponentiated sum of the clique potentials of the maximal cliques in the MRF, shown in Eq. 2.27.

A clique is just a fully-connected subgraph, whereas a maximal clique is one which will cease to be a clique if any other vertex, or node, is added to it. $V_c(x)$ is the clique potential. In a four-connected neighborhood the maximal clique is of the second order (an edge) and can also be called a pairwise potential. Such a potential is the Potts model, shown as follows:

$$V_{pq} = -\delta(p,q) = \begin{cases} -1, & \text{if } p = q \\ 0, & \text{otherwise} \end{cases} \tag{2.26}$$

where p and q are neighboring pixel labels, and $\delta(p,q)$ is the Kronecker-delta function, which is 1 when $p = q$ and 0 otherwise.

The probability that a MRF occupies a state $\mathbf{x}$ is written as:

$$P(X = x) = \frac{1}{Z} e^{-\sum\limits_{c \in C} V_c(x)} = \frac{1}{Z} e^{-E(x)} \tag{2.27}$$

where the Z function consists of the sum of the probabilities of all possible label assignments. It is a normalizing constant, or a partition function, ensuring that the distribution is a proper probability density function namely, that $\sum\limits_{x} P(X = x) = 1$. The right-hand side of the equation is written in terms of an energy function $E(x)$. Thus, a state that maximizes the probability $P(X = x)$ is the one that minimizes the energy $E(x)$.

It should be noted, however, that we have not defined any conditional probabilities of the labels, given some data, over the nodes of the graph. To minimize the aforementioned energy function, we can just label every node with the same value and achieve a trivial minimum. What is needed is a way to incorporate the unary terms that encode the probability of measuring the observation made at the node, given it is at a particular state. The resulting model is called a Hidden Markov Random Field, since we are trying to infer the unknown (hidden) labels. Alternatively, we can use CRFs, or Conditional Random Fields, that also try to reach the same goal, albeit in a different way.

2.3.5 HMRF vs. CRF

There are two key differences between a HMRF and CRF model, and it has to be noted that in practice, the term MRF is commonly used to refer to the hidden Markov random field. The first difference is that, in a HMRF, given the hidden labels, the observations are independent of each other, whereas in a CRF no such assumption is made. The second difference is, in essence, the broad divide between generative and discriminative models. A HMRF is a generative model, which means it tries to model the joint probability distribution:

$$P(X = x, Z = z) = P(Z = z|X = x)P(X = x) \tag{2.28}$$

whereas a CRF does not model the prior $P(X = x)$ and makes no assumptions about it. Instead, it directly models the conditional distribution $P(X = x|Z = z)$,

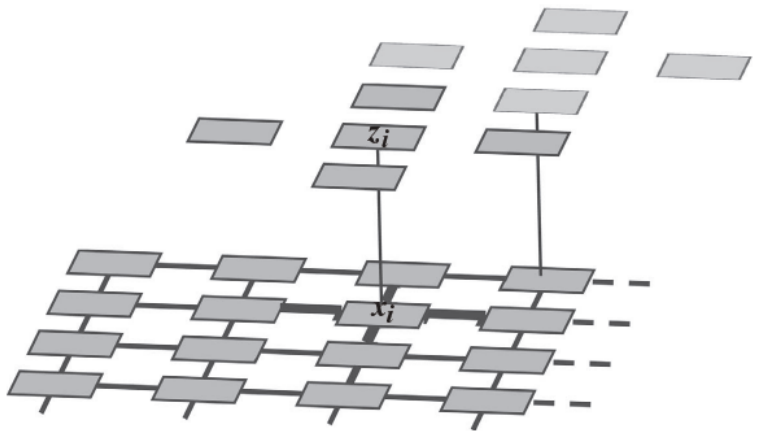

Figure 2.7: A graphical representation of the HMRF model, where z_i are observations and x_i are labels at a site i.

since that is enough to make a prediction.

Hidden Markov Random Field

HMRFs are Hidden Markov Models that operate on MRFs instead of Markov chains; they are widely used in image segmentation, where a structured output is desired. The usefulness of an HMRF lies in its ability to simultaneously enforce data faithfulness and spatial smoothness. We may think of the image labels as hidden variables which need to be inferred, while only having access to the noisy observations of nodes (pixels) like color and texture. Such a representation can be seen in Fig. 2.7, where Z denote observations and X the hidden variables.

As can be seen in Fig. 2.7, the MRF is comprised of the hidden variables X, analogous to the HMMs, and each node has an observation Z, which in our case are color and texture features.

The figure shows that any observation Z_i is independent of the others, given its hidden variable X_i, in direct analogy with HMMs. The posterior of an HMRF is defined as:

$$P(X = x|Z = z) \propto P(Z = z|X = x)P(X = x), \tag{2.29}$$

but there is usually a dependence on parameters θ. Therefore, a more formal definition is the following:

$$P(X = x|Z = z, \theta) \propto P(Z = z|X = x, \theta)P(X = x, \theta). \tag{2.30}$$

Finding the most likely configuration is done by a maximum a posteriori estimation (MAP) in the form of:

$$\begin{aligned} x^* = \arg\max_x P(Z = z|X = x, \theta)P(X = x, \theta)) &= \\ \arg\max_x P(X = x|Z = z, \theta) &= \\ \arg\min_x E(X = x|Z = z, \theta) & \end{aligned} \tag{2.31}$$

Therefore, minimizing the energy function maximizes the probability. A very popular energy function is the combination of a data term and smoothness term, formally defined as follows:

$$E(X = x, Z = z) = \sum_{i \in \mathcal{N}} -D(Z = z_i|X = x_i) + \sum_{(i,j) \in \mathcal{E}} V_{ij} \tag{2.32}$$

where the pairwise potential is the same as in Eq. 2.26 and the data term $D(Z = z_i|X = x_i)$ is just the negative likelihood that the observation z_i belongs to x_i.

The data constraint, or data faithfulness, is in essence the conditional probability, since we want the labels to adhere to the data. The prior is the smoothness term, which encourages neighboring sites to have the same labels, since it results in a lower energy.

Conditional Random Field

Where a HMRF models the joint probability $P(X = x, Z = z) = P(Z = z|X = x)P(X = x)$, we notice that we do not actually have to compute it, since the conditional probability $P(X = x|Z = z)$ would suffice for the purpose of labeling the sites, as is required in segmentation applications. This means finding the configuration of the hidden labels X, given Z. Therefore a CRF maximizes the following

$$x^* = \arg\max_x P(X = x|Z = z, \theta) = \arg\min_x E(X = x|Z = z, \theta), \tag{2.33}$$

whereas a HMRF maximizes the joint distribution, shown in Eq. 2.31. The two models are very similar, the only key difference being that CRF does not model the prior $P(X = x)$. The key is to use HMRFs when we can model it well, and CRFs otherwise. In this thesis, we use a CRF, since we can not say that two observations are independent, given their hidden labels, but the distinction between the two models must be made. This is complicated by the fact that many HMRF models are referred to MRF models, and the fact that CRFs and HMRFs can both be estimated with the same iterative algorithm, used in our work.

This thesis, however, has the word "autonomous" in its title, meaning unsupervised segmentation. This means that we can not compute the most likely segmentation since user interaction is not required and prior parameters on the number of components in the image are unknown. What we can do is apply a well-known iterative algorithm that attempts to present us with the most likely labeling as well as estimate the unknown parameters (since they are needed to find the most likely segmentation). This algorithm is described next.

2.3.6 Expectation Maximization

Expectation maximization (EM) is a method, famously presented in [14], that is used for finding the maximum a posteriori, or MAP, estimates of the model parameters. It is an iterative method that alternates between estimating the state likelihoods and the model parameters.

The motivation stems from the chicken-and-egg nature of this problem. If we have the parameters, it is easy to estimate the hidden variables by maximizing the log likelihood. Conversely, if we know the hidden variables, the parameters are easy to infer, as can be seen from the example below. Therefore, an iterative procedure, not unlike Newton's method, should yield successively better likelihoods using improved estimates of the parameters at each step.

The convergence for the method is proven in [47], and each step is guaranteed to increase the likelihood, but the result may be a local maxima. If we define $log\mathcal{L}(\theta \,|\, X, Z) = P(X, Z \,|\, \theta)$ to be the log-likelihood function, and E the expectation function, then the two steps may be written as follows:

$$Q(\theta|\theta^{(t)}) = E[\, log\mathcal{L}(\theta \mid X, Z) \mid Z, \theta^{(t)} \,], \tag{2.34}$$

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta|\theta^{(t)}), \tag{2.35}$$

where, as before, Z and X denote the observations and hidden variables, respectively, and $\theta^{(t)}$ are the estimated parameters at step t. Thus, the E step (2.34) consists of calculating the expectation of the data log-likelihood $\mathcal{L}$, and the M step (2.35) consists of finding the parameters θ that maximize this function.

2.4 Support Vector Machine

Support vector machines (SVM) are supervised discriminative models known for utilizing high-dimensional feature spaces without losing predictive power. In their most basic form, they are linear, binary and non-probabilistic classifiers, but there exist extensions, shown in the following paragraphs. The most simple way to define a SVM is to say that it tries to find a hyperplane between the (two) classes, usually denoted $y_i \in \{1, -1\}$, such that any data points are as far as possible from it. If our data has N features, we can think of it as an N-dimensional space and, by definition, the hyperplane is a $N-1$-dimensional subspace of our feature space.

We can also state the purpose of SVM as maximizing the margin, which is the distance between the hyperplane and the closest data point to it. An illustrative example is Fig. 2.8, where the two-dimensional feature space is composed of X_1 and X_2, and contains the hyperplane (which in this case is a line) and the color-coded class labels. Any hyperplane can be defined as follows:

$$w \cdot x - b = 0 \tag{2.36}$$

where x is the set of points that satisfy the equation, w is the vector perpendicular to the hyperplane, and $\frac{b}{||w||}$ is the distance along the vector w from the origin to the

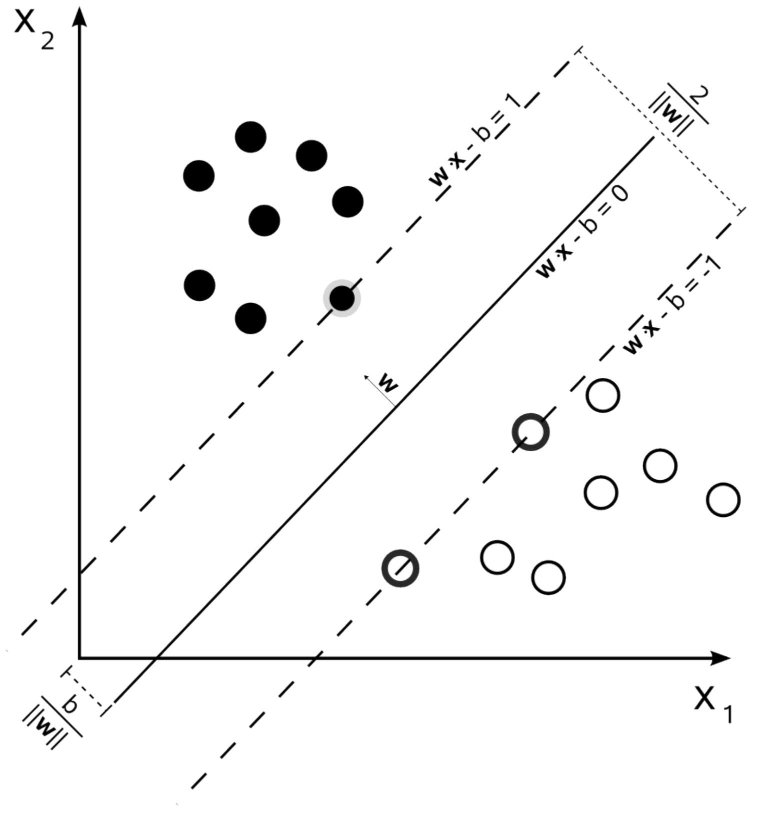

Figure 2.8: A two-dimensional representation of a SVM, the separating hyperplane, and its margin. Black labels denote the positive class and white labels the negative. Image from [13].

hyperplane. If the data is linearly separable, we want to maximize the distance $\frac{2}{||w||}$ between the following two hyperplanes:

$$w \cdot x - b = 1 \tag{2.37}$$

$$w \cdot x - b = -1 \tag{2.38}$$

This means we need to minimize $||w||$, formally expressed in Eq. 2.39, with the constraint shown in Eq. 2.40:

$$\arg\min_{w} \frac{1}{2}||w||^2, \tag{2.39}$$

$$y_i(w \cdot x_i - b) \geq 1, \tag{2.40}$$

for all data points $i \in 1,..,N$. The basic version of an SVM requires the setting of only a single parameter, and only in the case where the data are not linearly separable. In this case the "best" hyperplane can still be found by introducing slack variables λ_i for each data point, and Eq. 2.40 now becomes:

$$y_i(w \cdot x_i - b) \geq 1 - \lambda_i \tag{2.41}$$

Consequently, Eq. 2.39 becomes:

$$\arg\min_{w}\{\frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\lambda_i\} \tag{2.42}$$

where C is the regularization parameter, which controls the trade-off between maximizing the margin and minimizing the error. Finding the optimal w for both equations is a problem which is easily solved by quadratic programming techniques.

We can also modify SVMs to yield a solution with a non-linear hyperplane by applying the kernel trick, which transforms the feature space where the seemingly non-linear hyperplane is actually linear. This can be done by replacing the dot products $a \cdot b$ with non-linear kernel functions $k(a, b)$. A popular kernel is the following Gaussian radial basis function:

$$K(a, b) = e^{\frac{-\gamma||a-b||^2}{2}} \tag{2.43}$$

where γ is the parameter which controls the radius of the kernel.

Multiclass labeling is easily performed by using multiple binary classifiers. We can either use a one-versus-one or a one-versus-all approach. The former trains a binary classifier between each label, which yields $\frac{n(n-1)}{2}$ labels, where n is the number of labels. The latter trains n classifiers, that distinguish between label $i \in 1, ..n$ and the rest. In this thesis we used the one-versus-all approach, since it results in significantly fewer classifiers, while yielding comparable results.

Finally, SVMs can be made to output probabilities using Platt scaling [30]. If we assume that the labels $+1$ and -1 are given by some sign function $y = sign(f(x))$, whose output is a real-valued function $f(x)$, then we can produce a probability estimate $P(y = 1|x)$ as follows:

$$P(y = 1|x) = \frac{1}{1 + e^{Af(x)+B}} \tag{2.44}$$

where A and B are scalar parameters learned from the training set using cross-validation.

Now we have arrived at a probabilistic, non-linear, multilabel SVM classifier. This thesis uses the probabilistic output as the likelihood $p(y|x) = \mathcal{L}(x|y)$, computed for each label. We apply a Gaussian radial basis function, since it has improved the results. The parameters C and γ where optimized by cross-validation on a separate dataset. The next section explains how all of the previous theory fits into the algorithm and is followed by the results of this method.

Chapter 3

The proposed segmentation approach

The goal of this thesis is to succesfully partition images into a small number of segments, without any user input. To achieve this goal, we employ all of the methods in the previous chapters combined into an unsupervised iterative segmentation algorithm.

For our purposes, pixel-level representation is redundant, because the color and texture of a pixel is, more often than not, the same as its neighbors'. To reduce this redundancy, we apply a preprocessing step on the image that consists of oversegmenting it into several hundred superpixels, where a superpixel is a region in the image that consists of similar pixels. This is achieved by using the SLIC algorithm (Sec. 2.1).

The mathematical model behind the structured information in the image is a conditional random field (Sec. 2.3), where the nodes are superpixels and edges connect the superpixels that share a boundary. Producing a segmentation means inferring the hidden label associated with each node. Every node also has some observation Z, which is the information on which we base our segmentation. To produce the best segmentation, we need to maximize the following conditional probability:

$$x^* = \arg\max_x P(X = x|Z = z, \theta) \propto \arg\min_x E(X = x|Z = z, \theta), \tag{3.1}$$

where, as before, X denote the hidden labels, Z the observations, and θ the parameters. The goal is to minimize the energy function:

$$E(X = x|Z = z, \theta) = \sum_{i \in \mathcal{N}} \Psi_i + \sum_{i,j \in \mathcal{E}} \Phi_{ij}, \tag{3.2}$$

where Ψ denotes unary potentials, or the data likelihood term, and Φ is a pairwise potential between two neighboring superpixels. We definte the unary potentials as follows:

$$\Psi_i = -P(X_i = x_i|Z_i = z_i)P(X_i = x_i), \tag{3.3}$$

where $P(X_i = x_i)$ is the prior probability that the node i has the labeling x_i. The pairwise potential encodes the data smoothness, and is dependent on the color similarity between the two superpixels:

$$\Phi_{ij} = -P(X_i = x_i, X_j = x_j|Z_i = z_i, Z_j = z_j), \tag{3.4}$$

To obtain the data likelihood for each superpixel $P(X_i = x_i|Z_i = z_i)$ the visual observations Z are needed, which are some features computed for each superpixel. For this purpose, we use COLOR CHILD (Sec. 2.2), which is a descriptor that computes color and texture features of a region (in our case, the pixels that comprise the superpixel). We obtain a feature vector for each superpixel. We noticed that it is beneficial to decorrelate the features in the feature vector and we apply principal component analysis (PCA) on the feature vectors. It also reduces overfitting by removing noise and features that are uncorrelated with the labels. As can be seen in Fig. 3.1, with only a few dimensions we can accurately portray the data.

Each visual observation at a node (superpixel) is now a d-dimensional feature vector and we need to assign a label to each superpixel using these features. This is achieved by using a set of discriminative models (classifiers). In general, there are K classifiers. In this thesis we use SVMs 2.4, but the model is more general and any classifier can be used. Each classifier has its own parameters. However, both the superpixel labels and the parameters are unknown. To overcome this problem, we use expectation maximization (EM), shown in Sec. 2.3.6, to simultaneously estimate the labels and the parameters in each iteration until convergence.

In each iteration, we obtain the data likelihood $P(X_i = x_i|Z_i = z_i)$ by training K classifiers, one for each label. We use a one-versus-all approach, such that for each

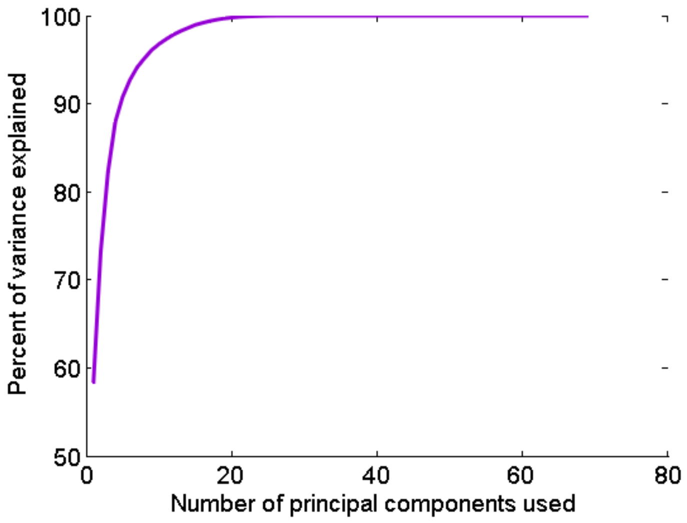

Figure 3.1: Plot of fraction of variance explained by the first d principal components.

label i, the superpixels that are currently labelled as such are the positive examples, and all the rest are negative examples. On the other hand, we have no information about the prior $P(X_i = x_i)$, which is uniform for all superpixels and labels in the first iteration, since we do not know the parameters. After each iteration it is updated as shown in Eq. 3.7. The unary potential is finally obtained by combining the prior with the probabilistic output from the SVM, as shown in Eq. 3.3.

To calculate the pairwise potentials we compute the probability of a superpixel i for each label, given its neighbors' labels and the edge weights connecting them. The following equation formally expresses the probability:

$$P(X_i = x_i | X_N = x_N, Z_N = z_N) = \sum_{j \in N} w_{ij} P(X_j = x_j | Z_j = z_j), \tag{3.5}$$

where X_N is the set of neighboring superpixels and w_{ij} is the weight of the edge between the two superpixels i and j computed as follows:

$$w_{ij} = e^{-\frac{1}{2}(d_{YCbCr} + d_{xy})}, \tag{3.6}$$

where the values in the color and coordinate distance measures, d_{YCbCr} (2.1) and d_{xy} (2.2) respectively, are the mean values of the pixels in each superpixel.

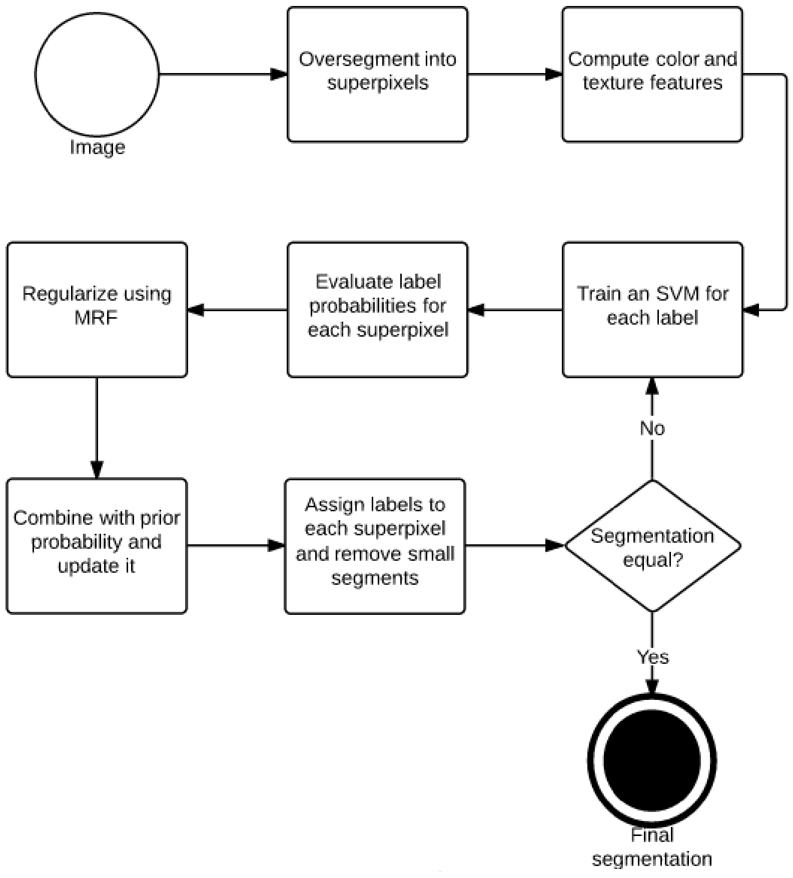
Image
Oversegment into superpixels
Compute color and texture features
Train an SVM for each label
Evaluate label probabilities for each superpixel
Regularize using MRF
Combine with prior probability and update it
Assign labels to each superpixel and remove small segments
Segmentation equal?
No
Yes
Final segmentation

Figure 3.2: A high-level overview of the segmentation algorithm.

Each iteration consists of the expectation (2.34) and the maximization (2.35) step of the EM algorithm. In the expectation step, we compute the likelihood of each label for each superpixel, given its observation. We obtain a K-by-N matrix, where K is the number of clusters (labels), and N is the number of superpixels. Next, we multiply the said matrix, the conditional probability $P(Z = z|X = x)$, with the K-by-N prior probability matrix $P(X = x)$, which yields the unary term.

To take into account the structural information we multiply the probabilities of the unary term with the pairwise term, computed as shown in Eq. 3.5 to obtain the final posterior probability. Lastly, the prior $P(X = x)$ is updated autoregressively with the posterior from each iteration as follows:

$$P(X = x) = P(X = x)P(X = x|Z = z). \tag{3.7}$$

In the maximization step, for each superpixel we choose the label with the highest posterior probability (MAP estimation) as shown in Eq. 3.1.

After each iteration, the number of labels is automatically reduced due to the nature of SVMs, which do not overfit the data if carefully parameterized. The number of labels after an iteration will remain the same only when each classifier distinguishes between its superpixels and all the rest reasonably well, and how well depends on the parameter C. Too large a value, and the labels are not merged, too small a value and labels which are not very similar are merged together.

The algorithm converges to a segmentation in about a dozen iterations, stopping when the difference in posteriors is too low. A high level overview can be seen in Fig. 3.2 and the pseudocode for the algorithm is shown in the Algorithm 3.

```
Apply SLIC superpixels;
Run COLOR CHILD on each superpixel to obtain features;
while not converged do
    //Expectation step;
    for each label i do
        Compute the likelihood of each data point belonging to class $i$;
        Combine the likelihood with the prior probability to obtain the
        data likelihood, using Eq. 3.3;
        Update the likelihood with the data smoothness term (Eq. 3.4)
        to get the posterior using Eq. 3.2;
        Update the prior using Eq. 3.7;
    end
    //Update step;
    Assign each superpixel to the most likely label using Eq. 3.1;
    for each label i do
        Remove if it contains no superpixels;
        Train an SVM classifier for the class $i$;
    end
end
```

Algorithm 3: The proposed segmentation algorithm.

Chapter 4

Experimental results

We have explained our segmentation approach in the previous section, but it needs to be compared with state-of-the-art algorithms on some database of images. The experiment is presented in this section. The parameters of our algorithm were fixed during comparisons and are shown in the following table: The top row explains to each algorithm the parameters refer. For SLIC, we have the trade-off in color and coordinate distance as m, the weight computation scale factor $scale$, which models the exponential function used for edge weights, and the number of superpixels N. In COLOR CHILD, we quantize the differential excitation into Q_{excit} values. Likewise, the fractional gradient orientation is quantized into Q_{orient} values. The value of the fraction is set to α, and importance of the color vs. texture features is controlled by β.

After applying PCA to our dataset, we take the first n components that together account for more than Var percent of the variance. For SVMs, we used the existing Matlab implementation libsvm [10], which requires setting the penalty value C and the flatness of the Gaussian kernel γ. The default value for C is the one in the

SLIC			COLOR CHILD				PCA	SVM	
m	$scale$	N	Q_{excit}	Q_{orient}	α	β	Var	C	γ
10	0.8	300	7	6	0.6	0.7	97	1	0.0001

Table 4.1: Parameter values.

table, and for all intents and purposes, the classification is linear, because the value of γ corresponds to a very flat Gaussian. We tried changing some of the parameters thought to be important, and presented the results in a later section.

The structure of this chapter is the following: we first describe the dataset and the performance measure in Sec. 4.1. Next, we present our experimental results compared to state-of-the-art image segmentation and boundary detection algorithms in Sec. 4.2, followed by an experiment on the sensitivity of our parameters in Sec. 4.3. Lastly, in Sec. 4.4 we visually analyze the quality of the segmentations, pointing out strengths and weaknesses.

4.1 Dataset and performance measures

Having a good segmentation algorithm is useful, but a comparison with state-of-the-art algorithms on some database of images is needed. Fortunately, there exists a segmentation evaluation database, first used in [2], that is suitable for the task. It is comprised of two datasets, which contain one and two objects in the foreground, respectively (and background). The database also offers the choice of intensity (grayscale) and color images, and we used the latter. A sample from both the one-object and two-object segmentation database can be seen in Fig. 4.1 and Fig. 4.2, respectively.

We chose this database over the Berkeley segmentation dataset [27] (BSD), because of the lack of ambiguity in the foreground objects, as opposed to the BSD, where human "ground truth" segmentations can differ wildly in the number of segments. In addition, segmentations on the latter database incorporate semantic cues, which are beyond the scope of any segmentation algorithm, especially those that use lower-level cues like color, texture and edges. Lastly, this dataset is especially biased towards soft-boundary-producing algorithms, since the evaluation computes the best threshold for hard segmentation instead of the algorithm choosing it. Conversely, we can see the agreement between human subjects in segmenting the foreground in the datasets of our choice in Fig. 4.1 and Fig. 4.2. The color of a pixel indicates it being labeled as foreground by one, two and three human subjects, with the colors red, green, and blue, respectively.

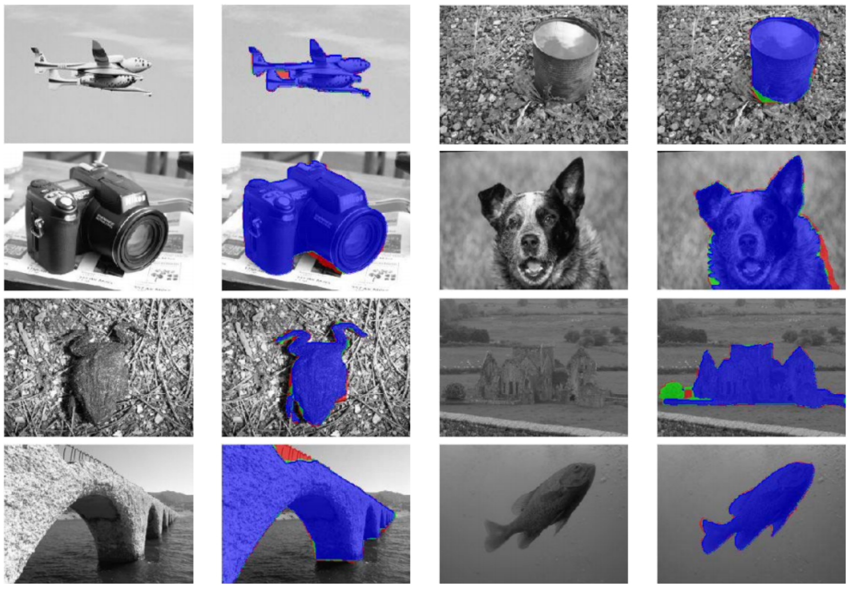
Figure 4.1: Some of the images in the one-object segmentation dataset and their corresponding ground truths.

The performance measure we used to evaluate the results is the F-measure as in [3]. It assesses the consistency of a segment with the ground truth and is defined as follows:

$$F = \frac{2PR}{P+R}, \tag{4.1}$$

where P and R denote precision and recall, respectively. Precision measures what fraction of the segment contains background, whereas recall measures the fraction of the foreground that is contained by the segment. First, we compute the F-measure for each segment individually and report the highest number. We also calculated the combined F-measure, which is chosen as the best value of a combination of segments covering the foreground object. It is computed as follows:

$$F_{multi} = \frac{1}{N}\sum_i F_i, \tag{4.2}$$

where an exhaustive search of the possible segment combinations is performed, and the highest average score is reported. Finally, we assess the fragmentation of the

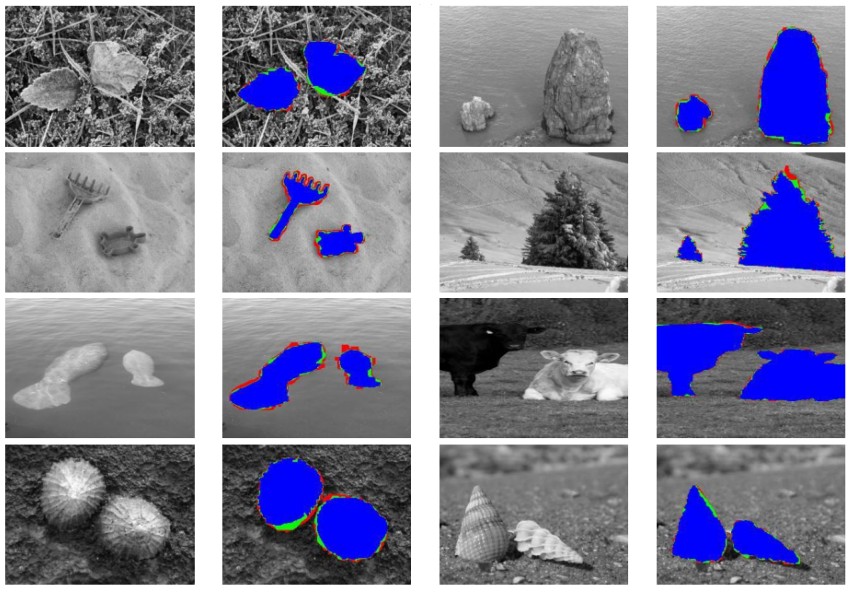

Figure 4.2: Some of the images in the two-object segmentation dataset and their corresponding ground truths.

object by counting the number of segments that comprise the combined F-measure. The exact equation is the following:

$$Frag_{object} = |1 - N|. \tag{4.3}$$

Lower fragmentation means that the foreground object is more correctly evaluated as being a single segment.

4.2 Experimental results

In addition to evaluating our approach on a segmentation dataset, we also need to compare the results to some other state-of-the-art techniques. The choice of algorithms for comparison was the same as in [3], namely:

- Probabilistic Bottom-Up Aggregation and Cue Integration [3], denoted by PBACI. It gradually merges pixels into successively larger regions by taking into account, intensity, geometry, and texture.

Algorithm	F-measure single	F-measure multi	Object fragmentation
Our method	0.71 ± 0.01	0.83 ± 0.01	$\mathbf{0.40} \pm 0.03$
PBACI	$\mathbf{0.86} \pm 0.01$	0.87 ± 0.02	1.66 ± 0.30
SWA	0.76 ± 0.02	0.86 ± 0.01	2.71 ± 0.33
N-cuts	0.72 ± 0.02	0.84 ± 0.01	2.12 ± 0.17
Gpb	0.54 ± 0.01	$\mathbf{0.88} \pm 0.02$	7.20 ± 0.68
Mean shift	0.57 ± 0.02	$\mathbf{0.88} \pm 0.01$	11.08 ± 0.96

Table 4.2: Results of single and multi-segment coverage on the one-object dataset (95% confidence).

- Segmentation by weighted aggregation [38], denoted by SWA, which determines salient regions in the image and merges them into a hierarchical structure.
- Normalized cuts [26], denoted by N-cuts. It treats the problem of segmentation by computing multiple minimum "normalized" cuts on a pixel graph.
- Contour detection and hierarchical Image Segmentation [6], denoted by Gpb, which reduces the problem to contour detection and uses spectral clustering to combine local cues into a global framework.
- Mean shift [11], a general mode-seeking algorithm on a non-parametric probability distribution, in this case, the color or intensity distribution.

The results in the following table present the average scores over all 100 images in the one-object dataset. The values for the other algorithms were already computed in [3], to which we add our method.

The results show that our method achieves the least fragmentation, while being competitive in the other two measures. As can be seen in Table 4.2, there is an inverse relationship between the two F-measures, which is explained by the fragmentation of an algorithm. Having high fragmentation results in oversegmenting the image, which means the foreground will be comprised of several segments. Thus, algorithms with a high fragmentation have low single-segment F-measure.

Conversely, having a high number of segments boosts the multi-segment F-measure, but the effect is not as pronounced. We believe that a good algorithm should foremost have low object oversegmentation, because it should depict the object as being comprised of a single segment.

4.3 Parameter sensitivity analysis

Even though our algorithm has a relatively high number of parameters, it is robust to changes to those parameters as can be seen in Fig 4.4. We have experimented with four parameters we thought were important and concluded that the algorithm is robust to changes in their values. More specifically, we varied the importance given to color as opposed to texture, the cut-off point for PCA, which is percentage of variance to be saved, the number of superpixels, and lastly, the value of the fractional derivative between 0 and 1. Because of the inherent capability of SVMs to deal with noisy and uncorrelated data, the results did not differ much. The parameters we did not test have the default values, proposed by the original authors.

4.4 Qualitative analysis

We can see some of the segmentations on the one-object dataset in Fig 4.3. For reference, we further present some successful and unsuccessful results of our segmentation algorithm on a few images from the BSD, in Fig. 4.5 and Fig. 4.6. We also evaluted the algorithm on a few images taken from a marine robotic boat, that is equipped with a RGB camera. The results can be seen in Fig. 4.7.

The advantage of our method is correctly delineating the object in the image as being comprised of a single segment. This is because similar superpixels are identified as having the same label early in the iterative process and we are only left with a few segments. The downside is that sometimes two distinct objects, or colors that are not too similar, are merged. There is also the problem of very small foreground objects as big as a single superpixel, which will be merged and lost in the early iterations.

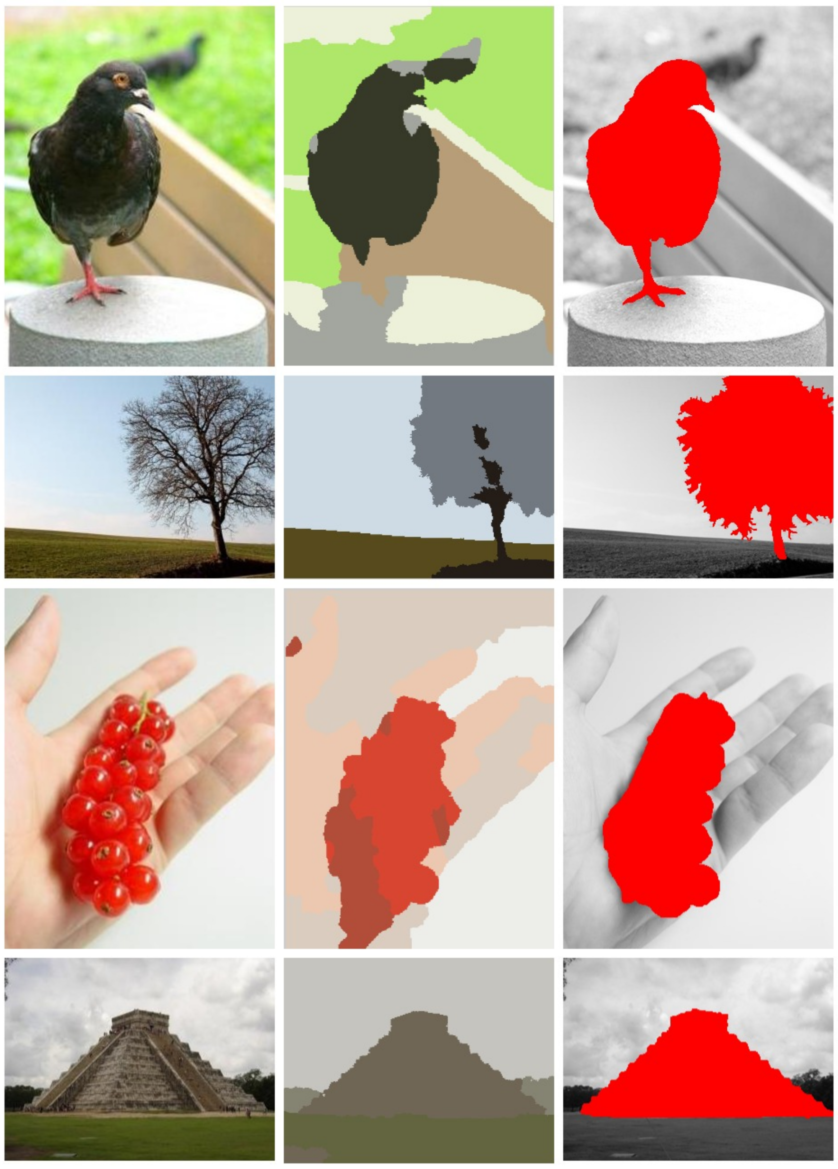

Figure 4.3: Images from the one-object dataset, our results, and the corresponding ground truths.

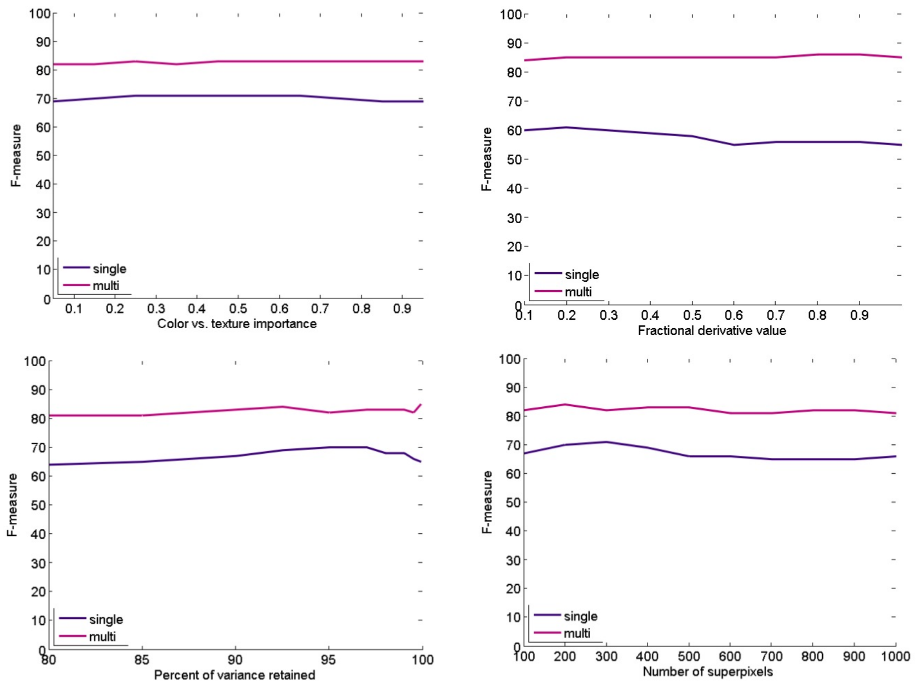

Figure 4.4: Clockwise from top left: COLORCHILD between-component ratio, texture component fractional derivative, PCA variance retained, and number of superpixels.

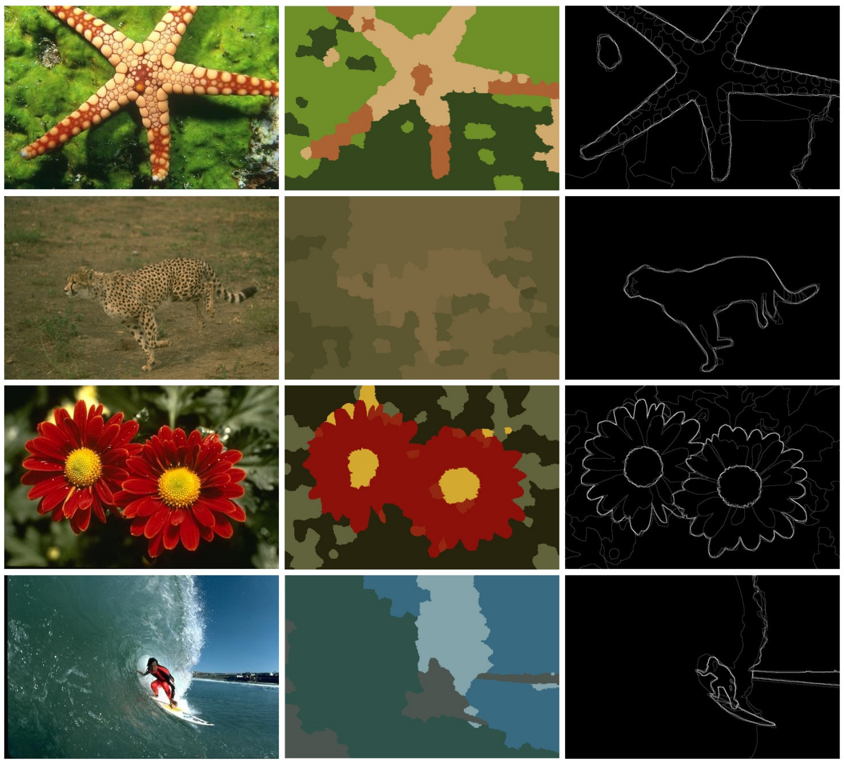

Figure 4.5: Images from BSDS, our results, and the corresponding ground truths.

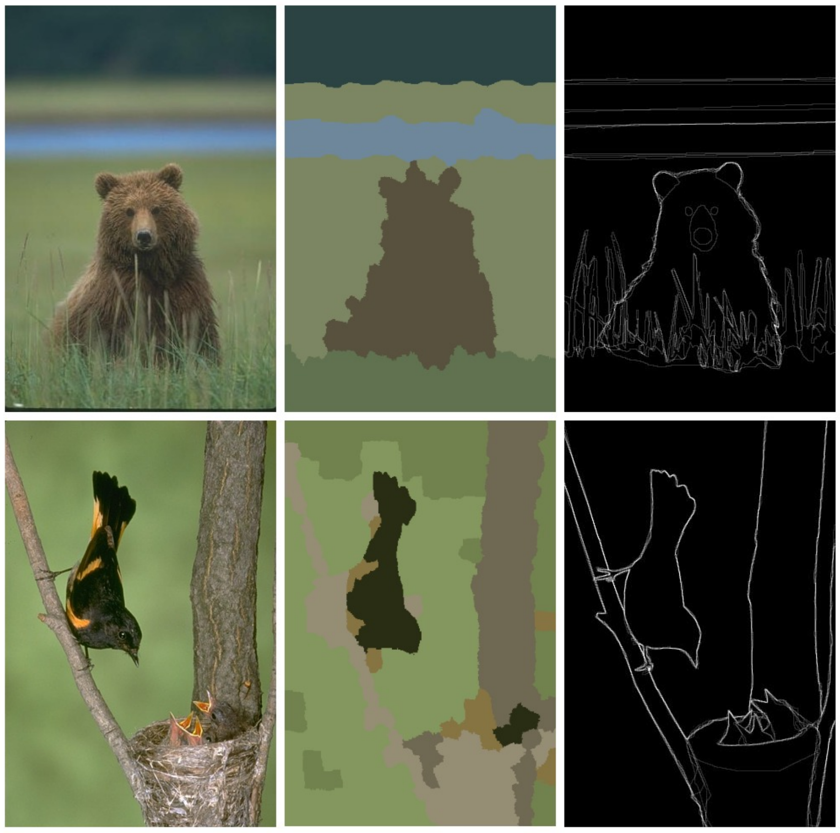

Figure 4.6: Images from BSDS, our results, and the corresponding ground truths.

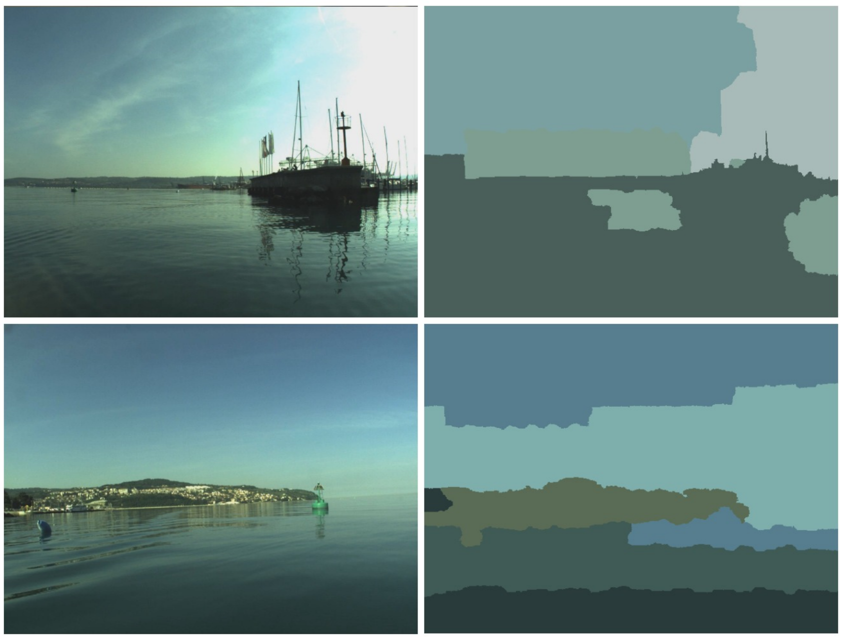

Figure 4.7: Image taken from the marine boat, and the corresponding segmentation.

Chapter 5

Conclusion

We developed an algorithm that segments an image without requiring user input. We aimed to obtain several coherent regions, similar to what a person might present if asked to segment the object out of the image. The algorithm begins with hundreds of small segments and iteratively reduces the number of labels until only several remain. We used an MRF to model the spatial coherence of the segmentation, and SVMs to assign labels to the superpixels, based on color and texture features.

The algorithm was tested on a segmentation evaluation database, and the results were similar to state-of-the-art algorithms. The F measure, which combines precision and recall, is comparable to other segmentation algorithms, but the fragmentation of our method, which measure the number of segments comprising each object, is by far the best. This means that we accurately judge an object by describing it with 1-2 segments instead of oversegmenting it like the other algorithms.

There are weaknesses, however, in that background superpixels that are very similar in color to the foreground object will be misclassified, even though they are separated. Such a deficiency could be fixed by a more careful choice of the data and smoothness constraint, as well as the interaction between the two, which leaves it open for future improvements.

In addition, future work would also involve a comparison of different classifiers. We chose SVMs for their robustness to overfitting in high-dimensional data. But other models like random forests [9], linear discriminant analysis, or Adaboost [19] might offer an improvement in accuracy. The labeling of the segments is also hard, so the option of soft-labeling segmentation, where a superpixel belongs to multiple labels simultaneously, can be explored. We can also use different functions for edge weights. In this thesis, the weight of the edge is modelled on color similarity, which leaves options like texture or boundary detection.

Of course, no single algorithm can generate the best results in every domain, let alone on every image. That's why it is important to incorporate domain knowledge and possible prior information which can help produce a better segmentation for a specific task. This thesis presents a general segmentation algorithm, applicable for any image, which can be modified and used to any domain-specific problem. It would therefore be interesting to specialize this method to domains like motion segmentation and interactive image segmentation.

Bibliography

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[2] S. Alpert, M. Galun, R. Basri, and A. Brandt. Image segmentation by probabilistic bottom-up aggregation and cue integration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[3] S. Alpert, M. Galun, A. Brandt, and R. Basri. Image segmentation by probabilistic bottom-up aggregation and cue integration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(2):315–327, 2012.

[4] S. H. Anamandra and V. Chandrasekaran. Color child: A robust and computationally efficient color image local descriptor. In *Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*, pages 227–234. IEEE, 2014.

[5] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2294–2301. IEEE, 2009.

[6] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[7] S. D. Babacan, R. Molina, and A. K. Katsaggelos. Generalized gaussian markov random field image restoration using variational distribution approximation. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1265–1268. IEEE, 2008.

[8] A. Blake, P. Kohli, and C. Rother. *Markov random fields for vision and image processing*. MIT Press, 2011.

[9] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[11] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[12] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[13] Cyc. Svm max sep hyperplane with margin. URL `http://commons.wikimedia.org/wiki/File:Svm_max_sep_hyperplane_with_margin.png`. [Online; accessed 9-September-2014].

[14] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[15] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE, 1999.

[16] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[17] M. A. Figueiredo and J. M. Leitao. Unsupervised image restoration and edge location using compound gauss-markov random fields and the mdl principle. *Image Processing, IEEE Transactions on*, 6(8):1089–1102, 1997.

[18] D. A. Forsyth and J. Ponce. *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference, 2002.

[19] Y. Freund, R. E. Schapire, et al. Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148–156, 1996.

[20] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[21] V. Garro, A. Fusiello, and S. Savarese. Label transfer exploiting three-dimensional structure for semantic segmentation. In *Proceedings of the 6th International Conference on Computer Vision/Computer Graphics Collaboration Techniques and Applications*, page 16. ACM, 2013.

[22] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 564–571. IEEE, 2013.

[23] J. M. Hammersley and P. Clifford. Markov fields on finite graphs and lattices. 1971.

[24] R. Kindermann, J. L. Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

[25] Z. Li, X.-M. Wu, and S.-F. 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.

[26] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International journal of computer vision*, 43(1):7–27, 2001.

[27] D. Martin, C. Fowlkes, D. Tal, and J. 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.

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

[29] R. Paget and D. Longstaff. Texture synthesis via a non-parametric markov random field. *Proceedings of DICTA-95, Digital Image Computing: Techniques and Applications*, 1:547–552, 1995.

[30] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*. Citeseer, 1999.

[31] M. O. Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[32] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[33] A. Rabinovich, S. Belongie, T. Lange, and J. M. Buhmann. Model order selection and cue combination for image segmentation. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1130–1137. IEEE, 2006.

[34] Z. Ren and G. Shakhnarovich. Image segmentation by cascaded region agglomeration. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2011–2018. IEEE, 2013.

[35] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[36] T. Ruzic, A. Pizurica, and W. Philips. Markov random field based image inpainting with context-aware label selection. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1733–1736. IEEE, 2012.

[37] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[38] E. Sharon, M. Galun, D. Sharon, R. Basri, and A. Brandt. Hierarchy and adaptivity in segmenting visual scenes. *Nature*, 442(7104):810–813, 2006.

[39] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[40] P. Sturgess, K. Alahari, L. Ladicky, and P. Torr. Combining appearance and structure from motion features for road scene understanding. 2009.

[41] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(7):787–800, 2003.

[42] K. A. Tran, N. Q. Vo, T. T. Nguyen, and G. Lee. Gaussian mixture model based on hidden markov random field for color image segmentation. In *Ubiquitous Information Technologies and Applications*, pages 189–197. Springer, 2014.

[43] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV 2008*, pages 705–718. Springer, 2008.

[44] A. J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13 (2):260–269, 1967.

[45] J. Wang, Y. Jia, X.-S. Hua, C. Zhang, and L. Quan. Normalized tree partitioning for image segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[46] X. Wang, H. Li, C.-E. Bichot, S. Masnou, L. Chen, et al. A graph-cut approach to image segmentation using an affinity graph based on l0- sparse representation of features. In *IEEE International Conference on Image Processing*, pages 4019–4023, 2013.

[47] C. J. Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

[48] J. Wu and A. C. Chung. A segmentation model using compound markov random fields based on a boundary model. *Image Processing, IEEE Transactions on*, 16(1):241–252, 2007.

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

[50] Z. Yang, F. Lang, X. Yu, and Y. Zhang. The construction of fractional differential gradient operator. *Journal of Computational Information Systems*, 7(12):4328–4342, 2011.

[51] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *Medical Imaging, IEEE Transactions on*, 20(1):45–57, 2001.