ABSTRACT

Title of dissertation: TOWARDS SEGMENTATION INTO SURFACES

Konstantinos Bitsakos,
Doctor of Philosophy, 2010

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science

Image segmentation is a fundamental problem of low level computer vision and is also used as a preprocessing step for a number of higher level tasks (e.g. object detection and recognition, action classification, optical flow and stereo computation etc). In this dissertation we study the image segmentation problem focusing on the task of segmentation into surfaces.

First we present our unifying framework through which mean shift, bilateral filtering and anisotropic diffusion can be described. Three new methods are also described and implemented and the most prominent of them, called Color Mean Shift (CMS), is extensively tested and compared against the existing methods. We experimentally show that CMS outperforms the other methods i.e., creates more uniform regions and retains equally well the edges between segments.

Next we argue that color based segmentation should be a two stage process; edge preserving filtering, followed by pixel clustering. We create novel segmentation algorithms by coupling the previously described filtering methods with standard grouping techniques. We compare all the segmentation methods with current state of the art grouping methods and show that they produce better results on the Berkeley and Weizmann segmentation datasets. A number of other interesting conclusions are also drawn from the comparison.

Then we focus on surface normal estimation techniques. We present two novel methods to estimate the parameters of a planar surface viewed by a moving robot when the odometry is known. We also present a way of combining them and integrate the measurements over time using an extended Kalman filter. We test the estimation accuracy by demonstrating the ability of the system to navigate in an indoor environment using exclusively vision.

We conclude this dissertation with a discussion on how color based segmentation can be integrated into a structure from motion framework that computes planar surfaces using homographies.

Towards segmentation into surfaces

by

Konstantinos Bitsakos

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2010

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Professor David Jacobs
Professor Amitabh Varshney
Dr. Cornelia Fermüller
Professor Jaydev P. Desai, Dean's Representative

Dedication

To my family and friends.

Contents

List of Tables

List of Figures

Chapter 1

Overview

In this dissertation we study the image segmentation problem focusing on the task of segmentation into surfaces. Arguably image segmentation is the most *important* low level vision task. Besides being by itself a very interesting signal processing problem, its importance also arises from the number of vision applications that require some sort of segmentation of the image. Object detection and recognition, face recognition, action classification, video and medical image analysis are a few of the domains that require a prior identification of "homogeneous" image regions. Moreover, other low level tasks, such as stereo and optical flow computation, greatly benefit from a good segmentation algorithm [1, 2].

A great number of researchers have extensively studied different variations of the segmentation problem with more or less success. As Borra and Shankar [3] suggest, the proper segmentation is task and domain specific. Hence, besides their *difficulty* as a high dimensional pixel grouping problems, most variations of the segmentation problem are also *ill-defined*. For example, when the goal is object recognition, image segmentation's purpose is to identify (and group together) image regions that correspond to objects. Since an object is not a well defined entity, this definition of image segmentation is also ill-defined. Furthermore, "correct" segmentations of an image may exist at different levels of detail, thus researchers have

(a) Coffee mug (b) Chair (c) Board game

Figure 1.1: Images of objects that are hard to segment into surfaces.

worked on "hierarchical" segmentation schemes [4, 5, 6].

We prefer a geometric based definition of segmentation that avoids most ambiguity problems mentioned above. More specifically, we use the surface normal of individual pixels as the criterion for grouping them together. According to this definition, areas with smooth surface normals should belong to the same segment, and segment boundaries should correspond to normal discontinuities, caused either by distance or orientation discontinuities.

This definition of the segmentation is straightforward in theory, but it presents many challenges on the algorithmic and implementation levels. In practice, it is impossible to compute the surface normal of an individual pixel unless a smoothness assumption about the region around the pixel is made. This leads to the well known chicken-and-egg problem, where one needs to assume that the area around a pixel possesses the same properties (similar surface normal in this case) as the pixel in order to compute those properties and check whether the properties of the

pixels around it actually have the same properties. In general, it is known that surface normal estimation belongs to the general category of structure from motion problems, that are harder than stereo and optical flow computation, since one seeks to estimate 3 dimensional quantities instead of 2D image properties. A very common assumption that we also adopt in this work is *the planarity assumption*, namely we assume that objects consist of planar surfaces.

Apparently there are important unresolved issues when surfaces are not planar, as in the case of the coffee mug or the chair of Fig. 1.1. It is not clear how a "proper" segmentation into surfaces algorithm should handle the smooth surface normal change. One might argue that the coffee mug should be considered as a single entity. What about the chair then? A division into two surfaces, one supporting the back and the other where one sits, perpendicular to each other seems a better solution than a single surface. In a sense the resolution of the surface normal estimation ultimately defines the segmentation. Even in cases when there is a clear surface normal discontinuity, such as the individual surfaces of the "mastermind" board game (Fig. 1.1), there are computational problems. In this particular example the edge between the two areas is much weaker than the texture edges on each individual segment. As a consequence any gradient based segmentation algorithm would fail to identify the edge. In general segmentation into planar surfaces is a very hard problem.

This dissertation does not claim to provide a complete solution to the segmentation into planar surfaces problem. A careful study of some parts of the whole system is performed, instead, and a number of improvements over current methods

are proposed. More specifically, we consider the subproblems of *color based segmentation* and *surface normal estimation* in isolation, and their *interaction*. Our two basic theses are that a) color-based segmentation should be treated as a filtering step followed by a grouping process and b) combination of curve based, region based and point based cues is important for surface estimation (and low level computer vision in general). In the next paragraphs we further develop these ideas and briefly describe the content of each chapter of this thesis.

We start, in chapter 2, by describing a framework through which mean shift, bilateral filtering and anisotropic diffusion can be described. The simplicity of the framework brings forth the similarities and differences of these methods resulting in a better understanding on how they operate on images. Furthermore, three new methods are described and implemented and the most prominent of them, called Color Mean Shift, is extensively tested and compared with the existing methods. Using a number of images and different performance criteria we conclude that Color Mean Shift outperforms the existing methods i.e., creates more uniform regions and retains equally well (or better) the edges between segments, while it is slightly slower than the existing methods.

Chapter 3 describes and experimentally verifies the thesis that color based segmentation should be a two stage process, namely an edge preserving filtering followed by a clustering of the image pixels. We create novel segmentation algorithms by coupling the filtering methods of the previous chapter with four clustering methods; connected components grouping with constant threshold in $3D$ or $5D$ space, grouping using region adjacency graphs; and the popular grouping using adaptive

threshold algorithm by Felzenszwalb and Huttenlocher [7]. Then, we use the Berkeley database to compare the segmentation results with those obtained from human subjects. We use a simple measure based on edge overlap as well as four popular measures to compare the quality of the segmentation. Extensive experimental comparison verifies that the two stage segmentation produces better results than any clustering algorithm in isolation. In addition, the results attests that the two stages are interconnected i.e., for best segmentation results the combination of filtering and grouping algorithms should considered together. Studying and improving an individual part (either filtering or grouping) does not guarantee better results. Appendix **A** presents more segmentation results using a different dataset obtained from the Weizmann Institute [8].

In the next chapter (4) we switch topic and focus on the surface normal estimation problem. More specifically, we describe how image cues can be combined with odometry (or inertial sensor) measurements to estimate the surface normal of image regions and perform visual navigation on a challenging indoor environment. We present one way to combine three different methods based on image points, straight lines and whole image regions and estimate the surface normal and distance of the walls more accurately and robustly. Besides the description of two novel methods for surface normal estimation based on straight lines and regions, this chapter also provides an paradigm on how an actual visual system can benefit from knowledge of the camera motion. In this case the odometry of the robot empowers us to a) decouple motion and structure and hence compute the surface normal using feature points by solving a linear system, b) estimate the surface normal by considering

the stretch of the whole region. In addition, we propose one way to integrate the measurements of the surface normal over time using an extended Kalman filter. The whole approach is implemented and tested on a mobile robot. In a number of experiments we demonstrate the ability of the system to navigate in an indoor environment using exclusively vision. The quality of visual navigation is used to evaluate the surface normal estimation with the individual methods and their combination. In all the experiments the combination of the three methods produces much better navigation results than each individual method in isolation. The integration of the surface normal measurements over time further improves the quality of the navigation.

Appendix B directly relates to chapter 4. The stretch filter that we develop was motivated by one of the surface normal estimation methods of that chapter, namely the harmonic shape from texture method. In a nutshell according to our method the surface normal and distance are encoded in the image stretch and shift of a planar region between two successive camera frames, thus by measuring the latter image quantities one can estimate the surface values. In this chapter we describe a direct way to estimate the stretch of a $2D$ signal using a properly created single filter. We analytically develop this filter and present results of applying it to real signals. We show that this method is a real-time alternative solution for measuring local signal transformations. Experimentally, this method can accurately measure stretch, however, it is very sensitive to shift.

Appendix D describes the process of calibrating the camera with respect to the Pan and Tilt Unit. This is a necessary procedure in order to use PTU based

measurements for the camera motion in structure from motion algorithms such as the ones used in chapter 4. First, we define what we mean by the term "calibration". Then, we formulate the calibration process as an optimization problem and describe its solution. Finally, we present the calibration results we obtained in our setting, namely a quad camera frame mounted on a PTU-46-17P70T pan and tilt unit by Directed Perception.

We conclude this dissertation, in chapter 5, by presenting a framework that incorporates color based segmentation into structure from motion algorithms. We focus on the problem of estimating the homography i.e., the transformation of the locations of points belonging to a 3D planar surface between two frames. We extend current approaches by obtaining an initial grouping of the feature points using our color based segmentation algorithm. Then, we compute the homographies using robust existing techniques and we further adjust the parameters of the segmentation based on the geometry of the scene. The latter step corresponds to the merging region step of traditional plane estimation algorithms. We also propose a splitting mechanism in regions where the reprojection error of feature points is large, based on color segmentation. Finally we briefly touch the problem of active segmentation into planar surfaces, but providing a lemma that can be used to predict the quality of the homography estimation. All the proofs for the lemmas used in this chapter are presented on Appendix C.

Chapter 2

A Framework for Filtering Algorithms

2.1 Introduction

This chapter and the next considers the problem of image segmentation, based only on the intensity values of an image. Color based segmentation is a fundamental and well studied problem in computer vision and many algorithms exist in the literature. The simplicity of this problem[1] as well as its direct connection to surface based segmentation make it an appropriate candidate for a starting point in our discussion.

We perceive segmentation as a two-step process; a *smoothing step* followed by a *grouping step*. The smoothing step attempts to bring closer intensities of neighboring pixels that belong to the same segment, while preserving (or even enhancing) the intensity difference across segment boundaries. The grouping step attempts to decide whether two neighboring pixels belong to the same segment or not. Arguably both steps are equally important, even though current methods only concentrate on one step of the process. Furthermore, their combination affects the final result.

First we study a number of smoothing techniques; the original mean shift [9] and its modified version[10, 11][2], bilateral filtering [12],[13], local mode filtering [14]

[1]Here we refer to the simplicity of the formulation of color based segmentation, namely group pixels with similar color properties together. We do not imply, though, that this problem is easy to solve or has been solved so far.

[2]In the recent papers, the original "mean shift" approach is called "blurring mean shift". We

and anisotropic diffusion [15]. We present all the above techniques as variations of a general optimization problem. Using such a formulation the similarities and differences between them are made clear. This framework also provides a natural way to classify them using two criteria. Using the classification criteria we propose three novel methods. Two of them (color mean shift and spatial mean shift) are variations of the mean shift filtering and the third one is an extension of bilateral filtering. Filtering experiments show that color mean shift actually outperforms mode finding in smoothing the images while preserving the edges.

2.1.1 Related Work

In this section we present related work on mean shift, since this is the main focus and motivation for the whole chapter. Following the success of Comaniciu and Meer's version of mean shift [11] the same basic algorithm for non parametric clustering has been used for object tracking [16], 3D reconstruction [17], image filtering [11], texture classification [18] and video segmentation [19] among other problems. The relatively high computational cost of a naive implementation of the method combined with the need for fast image processing led researchers to propose fast approximate variations of it. Most notably, two solutions for finding pairs of points within a radius have been proposed; the Improved Fast Gauss Transform based mean shift [20] for Normal kernels and the Locality Sensitive Hashing based mean shift [18].

Cheng [10] was the first to recognize the equivalence of mean shift to a step-

use a different name for the mean shift variant used in computer vision, namely "mode finding". So in the rest of this chapter the term **Mode Finding** (MF) refers to **Comaniciu and Meer's version of mean shift**.

varying gradient ascent optimization problem, and much later Fashing and Tomashi [21] showed that it is equivalent to Newton's method with piecewise constant kernels, and is a quadratic bound maximization for all other kernels. Yuan and Li [22] prove that mean shift is a half quadratic optimization for density mode detection when the profiles of the kernel functions are convex. Finally, Carreira-Perpinan [23] proves that it is equivalent to an EM algorithm when the kernel is Normal.

At the same time a number of extensions of the basic algorithm have been proposed. Shen et al. [24] and Yuan and Li [22] propose multi scale extensions to the original algorithm for detecting density modes at different resolutions. Extensions to general metric spaces were also developed [25, 26, 27, 28].

2.1.2 Notational Preliminaries

We represent the color image as a mapping $\mathbf{S}$ from the 2D space of the pixel coordinates to the 3D space of the intensity values (for color images). $\mathbf{x}_i$ is a 2D vector representing the spatial coordinates of pixel i ($i = 1 \ldots N$) and $\mathbf{S}(\mathbf{x}_i)$ is a vector that represents the three color channels. To simplify the notation we denote the intensities for a pixel $\mathbf{x}_i$ with a subscript, so $\mathbf{S}(\mathbf{x}_i) = \mathbf{S}_i$. We also denote the set of all pixels as $\mathtt{X}$ and the whole image $S(\mathtt{X})$. The cardinality of $\mathtt{X}$ is N.

In the following sections we use *bold letters* to represent *vectors* and the notation $[\mathbf{x}_i, \mathbf{S}_i]^T$ to indicate a concatenation of vectors. When we want to indicate the evolution of a vector over time we use superscripts, e.g. $[\mathbf{x}_i^0, \mathbf{S}_i^0]$ indicates the initial values of pixel $\mathbf{x}_i$ having intensity $\mathbf{S}_i$.

2.1.3 Kernel Functions

Definition(Kernel Function): Let $\mathbb{X}$ be a d-dimensional Euclidean space and $\mathbf{x} \in \mathbb{X}$. We denote with x_i the i^{th} component of $\mathbf{x}$. The L_2 norm of $\mathbf{x}$ is a non-negative number $||\mathbf{x}||$ such that $||\mathbf{x}||^2 = \sum_{i=1}^{d} x_i^2$. A function $K : X \rightarrow R$ is a kernel if and only if there exists another function $k : [0 \cdots + \infty] \rightarrow R$ such that

$$K(\mathbf{x}) = k(||\mathbf{x}||^2) \tag{2.1}$$

and

1. k is non negative
2. k is non increasing i.e.,

$$k(a) \geq k(b),\ if\ a < b \tag{2.2}$$

3. k is piecewise continuous and

$$\int_0^{+\infty} k(a)da < +\infty \tag{2.3}$$

Function $k(x)$ is called the *profile* of the kernel $K(\mathbf{x})$.

Often the kernel function is normalized i.e.,

$$\int_X K(\mathbf{x})d\mathbf{x} = 1. \tag{2.4}$$

Even though kernel functions are mostly used for kernel density estimation,

we use them in order to define optimization problems that we subsequently solve using standard gradient descent methods. Thus, we are not only interested in the kernel function $K(\mathbf{x})$ but also on its partial derivatives $\frac{\partial K(\mathbf{x})}{\partial \mathbf{x}}$. Next we define two kernel functions that we use; the Epanechnikov and the Gaussian kernel.

2.1.3.1 Epanechnikov kernel

The Epanechnikov kernel [29] has the analytic form

$$K_E(\mathbf{x}) = \begin{cases} c_E(1 - \mathbf{x}^T\mathbf{x}) & \mathbf{x}^T\mathbf{x} \leq 1 \\ 0 & otherwise \end{cases} \tag{2.5}$$

where $c_E = \frac{d+2}{2\pi^{d/2}}\Gamma(\frac{d+2}{2})$ is the normalization constant. Fig. 2.1(a) presents this kernel in the $1 - D$ case. The partial derivative of $K_E(\mathbf{x})$ with respect to element x_i of vector $\mathbf{x}$ is

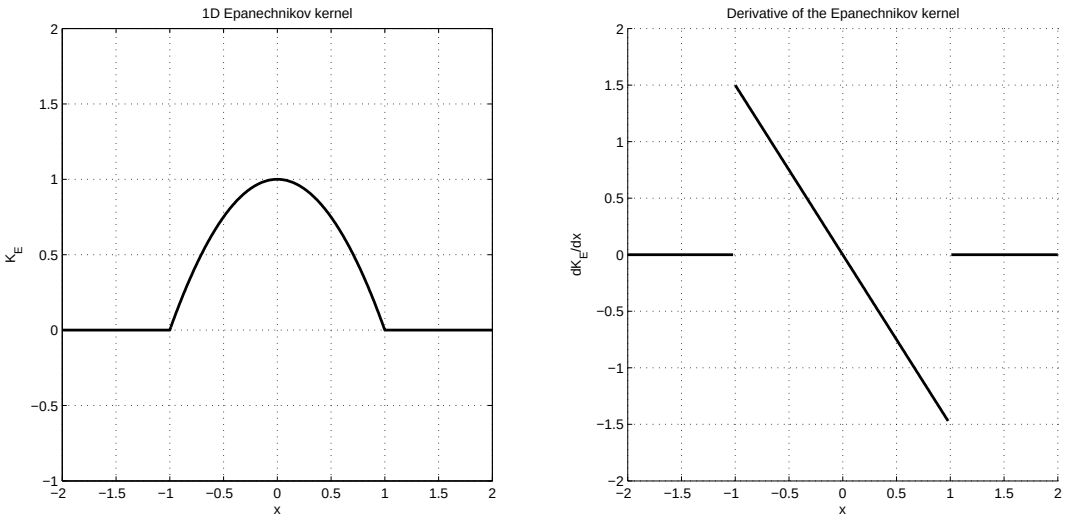
$$\frac{\partial K_E(\mathbf{x})}{\partial x_i} = \begin{cases} -2 \cdot c_E \cdot x_i & -1 < x_i < 1 \\ 0 & |x_i| > 1 \end{cases} \tag{2.6}$$

and is depicted in Fig. 2.1(b).

2.1.3.2 Multivariate Normal (Gaussian) kernel

The multivariate Normal kernel with variance 1 has the analytic form

$$K_N(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}). \tag{2.7}$$



(a) 1-D Epanechnikov Kernel (b) Derivative of 1-D Epanechnikov Kernel

Figure 2.1: $1-D$ Epanechnikov kernel.

In Fig. 2.2(a) a $1-D$ Normal kernel is displayed.

The partial derivative of $K_E(\mathbf{x})$ with respect to element x_i of vector $\mathbf{x}$ is

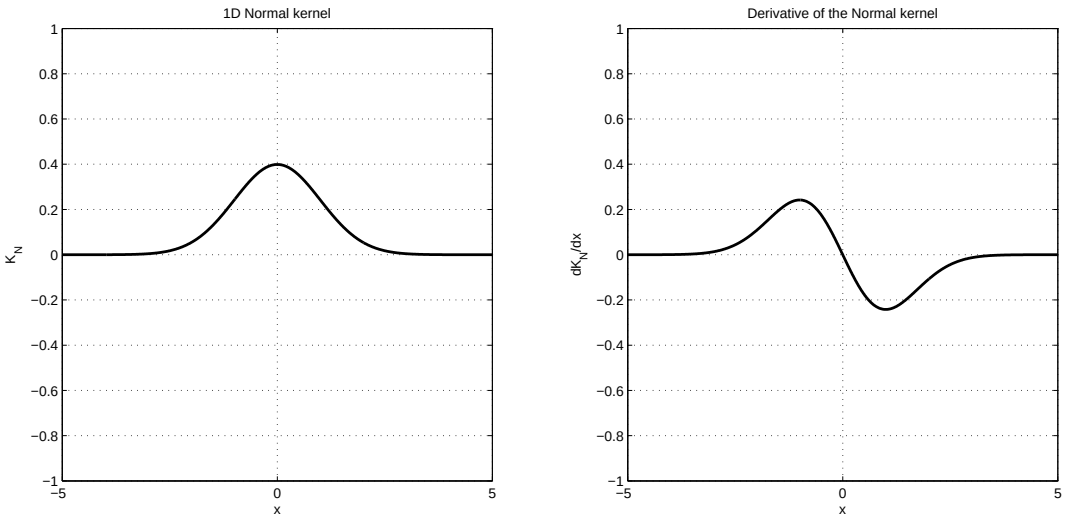
$$\frac{\partial K_N(\mathbf{x})}{\partial x_i} = -x_i \cdot (2\pi)^{-\frac{d}{2}} exp(-\frac{1}{2}\mathbf{x}^T\mathbf{x}) = -x_i \cdot K_N(\mathbf{x}) \tag{2.8}$$

and is depicted in Fig. 2.2(b).

The Normal kernel is often symmetrically truncated to obtain a kernel with finite support.

2.2 Image Filtering

In the following subsections we define a number of image filtering techniques as optimization problems. In previous formulations these methods were defined as the result of applying an algorithm to an image. Using our formulation we aim to



(a) 1-D normal kernel (b) Derivative of 1-D normal kernel

Figure 2.2: $1-D$ Normal kernel.

achieve two goals; to simplify the methods (since we only need a single equation to describe it) and to describe all the methods in a uniform way. Note that some methods (i.e. mean shift and mode finding) are defined for any kernel function, while others (i.e., bilateral filtering, local mode filtering and anisotropic diffusion) are only defined with respect to the Normal kernel $K_N(\mathbf{x})$.

2.2.1 Mean Shift (MS)

The original mean shift formulation [9] (applied to a color image) treats the image as a set of $5-D$ points (i.e., 2 dimensions for the spatial coordinates and 3 dimensions for the color values). Each point is iteratively moved proportionally to the weighted average of its neighboring points. At the end, clusters of points are formed. We

define mean shift to be the gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{S}_i]-[\mathbf{x}_j,\mathbf{S}_j]), \tag{2.9}$$

where $\sum_{i,j}$ defines the summation over all pairs of pixels in the image. Note that this problem has a global maximum when all the pixels "collapse" into a single point. We seek a local minimum instead. That's why we initialize the features $[\mathbf{x}_i,\mathbf{s}_i]$ with the original position and color of the pixels of the image and perform gradient descent iterations till we reach the local minimum.

2.2.2 Mode Finding (MF)

The modified mean shift formulation proposed by Comaniciu and Meer [11] (henceforth called "mode finding") can also be expressed as a gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i,\mathbf{S}_i]-[\mathbf{x}_j^0,\mathbf{S}_j^0]) \tag{2.10}$$

There is a subtle difference between mode finding and mean shift, that significantly affects the performance. In the former formulation each current point is compared against the original set of $5-D$ points $[\mathbf{x}_j^0,\mathbf{S}_j^0]$, while in the latter case the point is compared against the set of points from the previous iteration $[\mathbf{x}_j,\mathbf{S}_j]$. In a recent paper [30] S. Rao et al. study those two variations from an information theoretic perspective and conclude that mean shift is not stable and hence should

not be used for clustering.

Fig. **2.3** presents the results of both methods in a smoothly varying intensity image. Notice that the gradient of the kernel function is zero everywhere but in the boundaries. Thus, mode finding filtering only changes the intensity on the boundaries (that change is not very visible in Fig. 2.3). Mean shift, on the other hand, produces artificial segments of uniform intensity. Intuitively, each iteration of the process results in more clustered data which in turn results in better clustering results for the next iteration. On the downside, a fast mean shift implementation is challenging due to the fact that the feature points and the comparison points do not lie on a regular spatial grid anymore. Thus in a naive implementation one would have to compare the current feature $[\mathbf{x}_i, \mathbf{S}_i]$ against all the remaining feature points.

2.2.3 Spatial Mean-Shift (SMS)

One of our proposed methods that lies between mean shift and mode finding, spatial mean shift performs mean shift in the spatial dimensions and mode finding in the color dimensions. SMS can be viewed as the gradient descent solution of the optimization problem

$$\arg\min_{[\mathbf{x}_i,\mathbf{S}_i]} -\sum_{i,j} K([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j^0]). \tag{2.11}$$

Spatial mean shift suffers from the same computational problems as mean shift, so it is mentioned here for the sake of completeness. We exclude the results of both mean shift and spatial mean shift in our filtering and segmentation experiments.

2.2.4 Color Mean-Shift (CMS)

Color mean shift is our proposed method that alleviates the computational problem of mean shift by using the original spatial location of the points for comparison, while it uses the updated intensity values of the previous iteration for improved clustering ability. In a sense, mean shift is performed on the color dimensions and mode finding on the spatial dimensions (that is the reason for naming the method "color mean shift"). As above, CMS can be expressed as the gradient descent solution of the optimization problem

$$\arg \min_{[\mathbf{x}_i, \mathbf{S}_i]} - \sum_{i,j} K([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]). \tag{2.12}$$

2.2.5 Local Mode Filtering (LMF)

Local mode filtering [14] was introduced as a method to find the local mode in the range domain of each pixel of the image. A generalization of the spatial Gaussian filtering to a spatial and range Gaussian filter is used to iterate to the local mode (on the $3-D$ color domain). On each iteration the intensity of each pixel is replaced by a weighted average of its neighbors. From an optimization point of view the problem can be expressed as

$$\arg \min_{\mathbf{S}_i} - \sum_{i,j} K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]). \tag{2.13}$$

2.2.5.1 Bilateral Filtering (BF)

In bilateral filtering [12],[13] the intensity of each pixel is replaced by a weighted average of its neighbors. The weight assigned to each neighbor decreases with both the distance in the image plane (spatial domain) and the distance on the intensity axes (range domain). Formally the intensity at each pixel $\mathbf{S}_i$ takes the value

$$\mathbf{S}_i = \frac{\sum_j \mathbf{S}_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}. \tag{2.14}$$

Bilateral filtering can be considered *as the first iteration of local mode filtering with a specific step size* (Sec. *2.2.7*).

2.2.5.2 Joined Bilateral filtering

In this variation of the bilateral filtering both the intensity and position of each pixel is replaced by a weighted average of its neighbors. Formally, the new coordinates and color of each pixel are

$$[\mathbf{x}_i, \mathbf{S}_i] = \frac{\sum_j [\mathbf{x}_i, \mathbf{S}_i] K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0])}. \tag{2.15}$$

Analogous to bilateral filtering this method can be considered as the first iteration of mode finding with a specific step size.

2.2.6 Anisotropic Diffusion (AD)

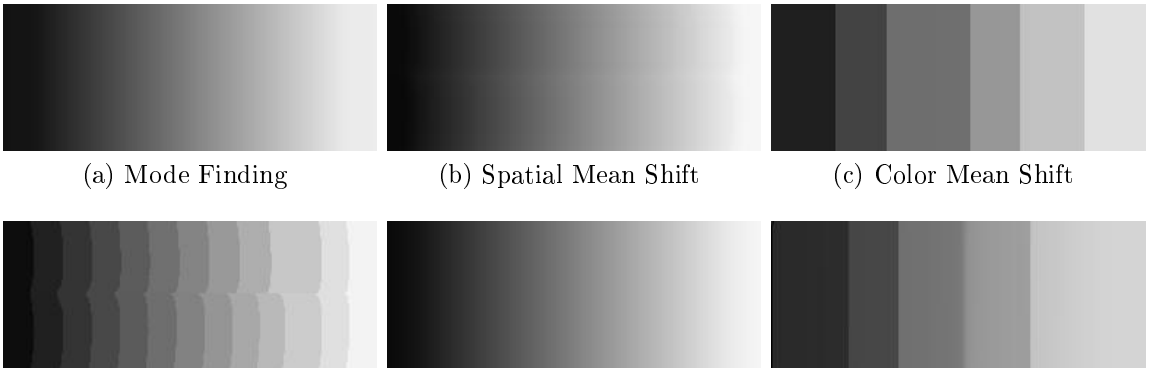
Anisotropic diffusion is a non-linear process introduced by Perona and Malik [15] for edge preserving smoothing. In the original formulation a diffusion process with a monotonically decreasing diffusion function of the image gradient magnitude is used to smooth the image while preserving strong edges. Since then other functions have been proposed and the equivalence of this technique to robust statistics has been established [31]. In [14] the connection with local mode filtering was also made. Here we provide an alternative view of the diffusion process as an optimization problem

$$\arg\min_{\mathbf{S}_i} -\sum_{i,j} K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j]). \tag{2.16}$$

The difference between this method and local mode filtering is analogous to the difference between the original mean shift and mode finding. Namely in local mode filtering the current point is compared against the original image pixels $[\mathbf{x}_j^0, \mathbf{S}_j^0]$, while in anisotropic diffusion the comparison is against the intensity value of the pixels in the previous iteration $[\mathbf{x}_j, \mathbf{S}_j]$.

2.2.7 Optimization steps sizes

From the above optimization problems *mean shift, spatial mean shift, color mean shift and anisotropic diffusion are joint optimization problems* i.e., the whole image needs to be optimized simultaneously. In mode finding and local mode filtering, on the other hand, each pixel can be optimized independently from the rest of the image. Next we present two claims concerning the step size of these optimization



(a) Mode Finding (b) Spatial Mean Shift (c) Color Mean Shift

(d) Mean Shift (e) Local Mode Filtering (f) Anisotropic Diffusion

Figure 2.3: All the described algorithms applied on a smoothly varying image. All the filtering algorithms were executed with spatial resolution $h_s = 21$ and range resolution $h_r = 10$ and used a Normal kernel.

problems.

Claim 2.1. Local mode filtering (and mode finding with a Gaussian kernel) can be considered as gradient descend methods for solving the corresponding optimization problem (Eqs. 2.13 and 2.10 respectively) with a step size at iteration t of

$$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])}. \tag{2.17}$$

Proof. A proof for local mode filtering follows. Each pixel p_i is optimized separately. So if we replace the step size γ_i in the general gradient descent algorithm we get

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \nabla \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0]) \tag{2.18}$$

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \sum_j \nabla K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0]) \tag{2.19}$$

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t - \gamma_i^t \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])[\mathbf{S}_j^0 - \mathbf{S}_i^t] \tag{2.20}$$

$$\mathbf{S}_i^{t+1} = \mathbf{S}_i^t + (\gamma_i^t \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j^0]))\mathbf{S}_i^t - \gamma_i^t \sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])\mathbf{S}_i^t \tag{2.21}$$

$$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])\mathbf{S}_j^0}{\sum_j K_N([\mathbf{x}_i, \mathbf{S}_i^t] - [\mathbf{x}_j, \mathbf{S}_j^0])} \tag{2.22}$$

that is exactly the intensity values for pixel $\mathbf{x}_i$ at the next iteration $t+1$. □

To prove the claim for mode finding with a Gaussian kernel one only needs to replace the occurrence of $\mathbf{S}_i^t$, $\mathbf{S}_i^{t+1}$, $\mathbf{S}_j^0$ with $[\mathbf{x}_i^t, \mathbf{S}_i^t]$, $[\mathbf{x}_i^{t+1}, \mathbf{S}_i^{t+1}]$, $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ respectively, because the optimization is performed on the $5-D$ domain.

Claim 2.2. Mode finding with an Epanechnikov kernel can be considered as a gradient descend method for solving the corresponding optimization problem (Eq. 2.10) with a step size at iteration t of

$$\gamma_i^t = -\frac{1}{2c_E \sum_{j,|||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]|||<1} 1} \tag{2.23}$$

As a consequence the result after one iteration of the gradient descent is

$$[\mathbf{x}_i^{t+1}, \mathbf{S}_i^{t+1}] = \frac{\sum_{j,|||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]|||<1} [\mathbf{x}_j^0, \mathbf{S}_j^0]}{\sum_{j,|||[\mathbf{x}_i^t, \mathbf{S}_i^t] - [\mathbf{x}_j^0, \mathbf{S}_j^0]|||<1} 1}. \tag{2.24}$$

Table 2.1 summarizes the optimization step sizes for each method along with

the results after one iteration. Note that in the case of mean shift and anisotropic diffusion we are using the *block gradient descent* method and optimize one pixel vector at a time[3].

2.3 Classification framework

Careful examination of the previous defined optimization problems reveal that there are only two differences in their objective functions; the presence of $[\mathbf{x}_i, \mathbf{S}_i]$ or $[\mathbf{S}_i]$ as the optimization argument; and the comparison against the points in the original image $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ or the points on the previous iteration $[\mathbf{x}_j, \mathbf{S}_j]$. Finally two of the methods (bilateral filtering and joined bilateral filtering) are an one-iteration methods, while all the other methods perform multiple iterations till convergence. Next we explain in details these differences, and define a classification of the methods based on these criteria.

2.3.1 $\arg\min_{[\mathbf{x}_i, \mathbf{S}_i]}$ vs $\arg\min_{\mathbf{S}_i}$

In the first case the optimization problem is defined over the joint spatial and range domain $(5 - D)$, i.e. both the position of the pixels as well as their intensities change in each iteration. In the second case, where the optimization is over the range domain $(3 - D)$, only the intensities of the pixels change while their position remain the same. This is not to be confused with the use of $[\mathbf{x}_i, \mathbf{S}_i]$ in the objective function. While the position of the pixel is always considered in the computation

[3]We use the symbols $\mathbf{x}_j$, $\mathbf{S}_j$ to denote the current value of pixel p_j. These might be the values of pixel p_j at iteration t or $t+1$ depending on whether p_j is processed after or before p_i.

Method	Step Size	Single iteration result
Mode Finding with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1}[\mathbf{x}_j^0,\mathbf{S}_j^0]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0]\|\|<1} 1}$
Mode Finding with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])[\mathbf{x}_j^0,\mathbf{S}_j^0]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$
Mean Shift with K_E	$\gamma_i^t = -\frac{1}{4c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1}[\mathbf{x}_j,\mathbf{S}_j]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j]\|\|<1} 1}$
Mean Shift with K_N	$\gamma_i^t = -\frac{1}{2\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])[\mathbf{x}_j,\mathbf{S}_j]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$
Spatial Mean Shift with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1}[\mathbf{x}_j,\mathbf{S}_j^0]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0]\|\|<1} 1}$
Spatial Mean Shift with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])[\mathbf{x}_j,\mathbf{S}_j^0]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j^0])}$
Color Mean Shift with K_E	$\gamma_i^t = -\frac{1}{2c_E \sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1} 1}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1}[\mathbf{x}_j^0,\mathbf{S}_j]}{\sum_{j,\|\|[\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j]\|\|<1} 1}$
Color Mean Shift with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])}$	$[\mathbf{x}_i^{t+1},\mathbf{S}_i^{t+1}] = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])[\mathbf{x}_j^0,\mathbf{S}_j]}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j])}$
Local Mode Filtering with K_N	$\gamma_i^t = -\frac{1}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$	$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])\mathbf{S}_j^0}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j^0,\mathbf{S}_j^0])}$
Anisotropic Diffusion with K_N	$\gamma_i^t = -\frac{1}{2\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$	$\mathbf{S}_i^{t+1} = \frac{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])\mathbf{S}_j}{\sum_j K_N([\mathbf{x}_i^t,\mathbf{S}_i^t]-[\mathbf{x}_j,\mathbf{S}_j])}$

Table 2.1: Step sizes and iteration results for the different filtering methods with different kernels.

Color Mean Shift

Input:
set of pixels $\mathbf{x}_i^0$ with intensities $\mathbf{S}_i^0$
a function g
Output:
feature vector $[\mathbf{x}_i, \mathbf{S}_i]$
Algorithm:
initialize feature points $[\mathbf{x}_i, \mathbf{S}_i] \leftarrow [\mathbf{x}_i^0, \mathbf{S}_i^0]$
repeat until convergence
for all features $[\mathbf{x}_i, \mathbf{S}_i]$

$$[\mathbf{x}_i, \mathbf{S}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{S}_j] g(|||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]||^2)}{\sum_j g(|||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j]||^2)}$$

Mode Finding

Input:
set of pixels $\mathbf{x}_i^0$ with intensities $\mathbf{S}_i^0$
a function g
Output:
feature vector $[\mathbf{x}_i, \mathbf{S}_i]$
Algorithm:
initialize feature points $[\mathbf{x}_i, \mathbf{S}_i] \leftarrow [\mathbf{x}_i^0, \mathbf{S}_i^0]$
for all features $[\mathbf{x}_i, \mathbf{S}_i]$
repeat until convergence

$$[\mathbf{x}_i, \mathbf{S}_i] \leftarrow \frac{\sum_j [\mathbf{x}_j, \mathbf{S}_j] g(|||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||^2)}{\sum_j g(|||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j^0, \mathbf{S}_j^0]||^2)}$$

Figure 2.4: The algorithms that we use in the experiments. Note that $g(x) = [x \leq 1]$ (indicator function in Iverson notation) for the Epanechnikov kernel and $g(x) = \exp(-x/2)$ for the Normal kernel. Local mode filtering is performed in a similar way as mode finding and mean shift, anisotropic diffusion are performed in a similar way as color mean shift.

of the objective function, that position might change or not (depending on the method).

At this point we should also make clear that the optimization is defined for the whole image, that is the values of all the pixels change. For the sake of simplicity we don't make this explicit when we write down the optimization equation.

2.3.2 $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ vs $[\mathbf{x}_j, \mathbf{S}_j]$

With a subscript we denote the value of the pixels at a specific iteration, so $[\mathbf{x}_j^0, \mathbf{S}_j^0]$ is the value of pixel $\mathbf{x}_j$ at the very beginning, i.e. in the original image. The lack of a superscript denotes the current value of pixels, i.e. the value of the pixel at a previous iteration. Two pairs of algorithms (mean shift/mode finding and local mode filtering/anisotropic diffusion) only differ in whether we compare the current value of a pixel against the original image or the image obtained in the previous iteration. As we will demonstrate in the experiments, the results vary significantly because of that (also see [30] for a theoretical analysis and justification).

Furthermore, there are two valid hybrid combinations that have not been proposed before.

- $[\mathbf{x}_j^0, \mathbf{S}_j]$: In this case the comparison is performed against the original position of the pixels and the previously computed range image.

- $[\mathbf{x}_j, \mathbf{S}_j^0]$: In this case the position of the pixels in the previous iteration is used along with their original intensity values.

Apparently the previous cases only make a difference when the optimization is de-

fined over the joint spatial/range domain. Otherwise the position of the pixels never changes, thus $[x_j] \equiv [x_j^0]$.

2.3.3 A taxonomy of filtering methods

Fig. **2.5** presents the various methods and where they fit with respect to the previous criteria. The three new methods are spatial mean shift, color mean shift and joined bilateral filtering.

2.4 Filtering experiments

Following the example of Comaniciu and Meer [11], we normalize the spatial and color coordinates of each pixel vector by dividing by the spatial (h_s) and color (h_c) resolution. Thus, the original feature vector $[\mathbf{x}_i, \mathbf{S}_i]$ is transformed to $[\frac{\mathbf{x}_i}{h_s}, \frac{\mathbf{S}_i}{h_r}]$ (not included in the optimization equations for simplicity reasons). Then, we perform the optimization; one pixel at a time in the case of mode finding (Fig. 2.4, top right), or one iteration of the whole feature set at a time in the mean shift and color mean shift cases (Fig. 2.4, top left). Fig. 2.6 displays the original images that we use for all the experiments in the rest of the section.

2.4.1 Epanechnikov vs Normal Kernel

First we present some filtering results when using different kernels; namely the Epanechnikov and Normal kernel (Figs. 2.7,2.8). Each column of the figures depicts the filtering result with a different algorithm; MF, LMF, CMS and AD stand

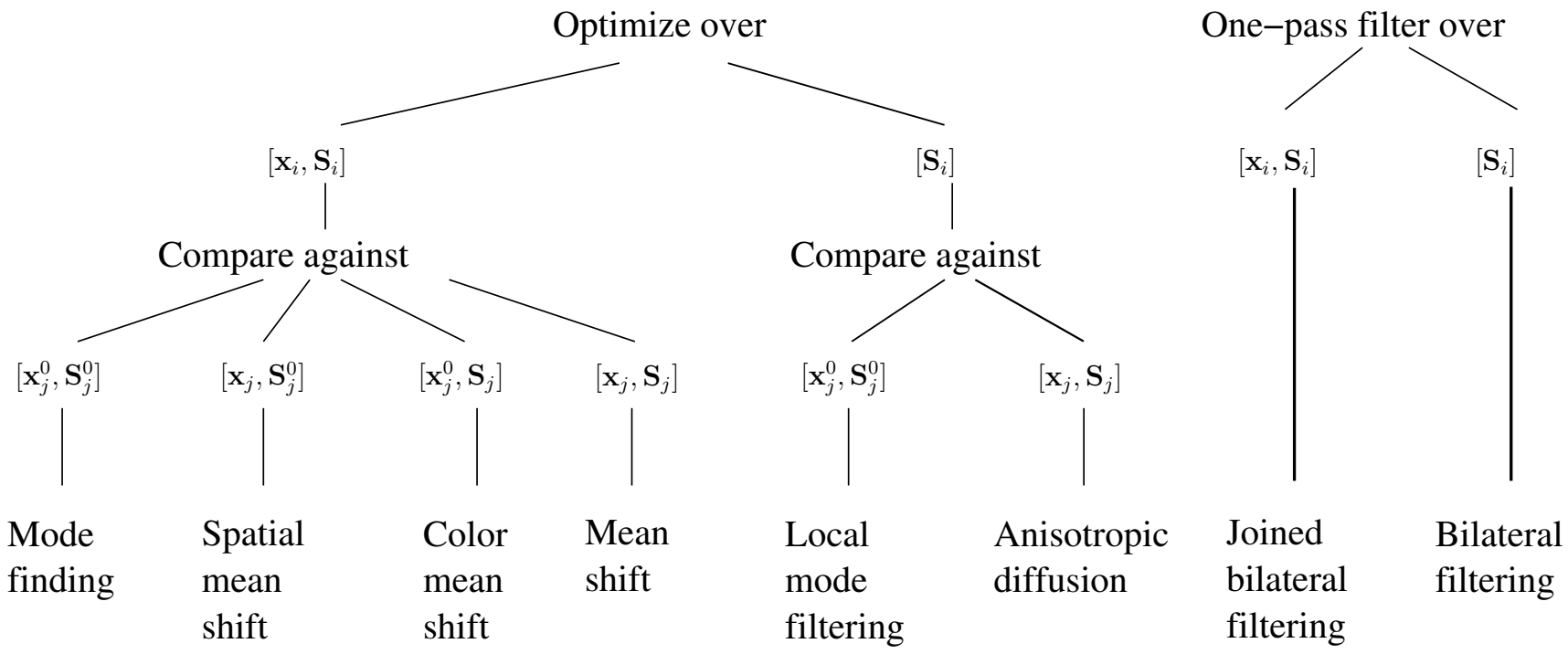


Figure 2.5: Classification of various filtering methods.

(a) Hand

(b) Workers

(c) Woman

(d) Houses

Figure 2.6: The original images we use for the filtering experiments. The first image is taken from Comaniciu and Meer's mean shift segmentation paper, while the remaining are training images of the Berkeley segmentation database collection. Their sizes are 303×243 and 481×321 pixels respectively.

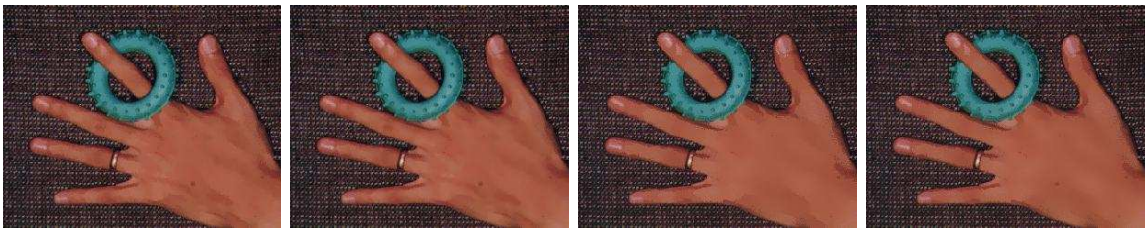
for mode filtering, local mode filtering, color mean shift and anisotropic diffusion respectively. In all cases the Normal kernel produces smoother results, while preserving edge discontinuities. As a matter of fact the color resolution h_r is the one that defines the gradient magnitude above which there is an edge (to be preserved). So for the "hand" image, a color range of $h_r = 19$ results in smoothing most of the texture on the background, while a value of $h_r = 10$ retains most the texture (in RGB color space with a Normal kernel).

In all the images mode finding and local mode filtering produced very similar results. Furthermore color mean shift and anisotropic diffusion gave similar results. Color mean shift seems to produce more crisp edges while anisotropic diffusion smooths some of the edges. Overall, color mean shift and anisotropic diffusion produce more uniform regions (e.g. suppresses the skin color variation on the "hand" image) and more crisp boundaries between segments compared to mode finding and local mode filtering. The latter is particularly important for the segmentation step. We further investigate this phenomenon in subsection 2.4.3.

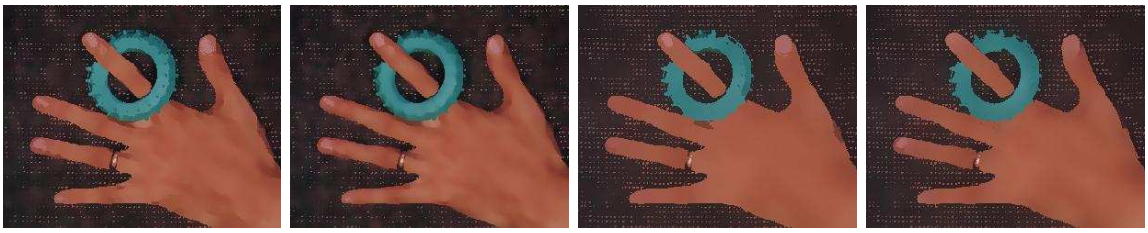
For the remaining filtering experiments we use a Normal kernel.

2.4.2 RGB vs Luv Color Space

In Figs. 2.9, 2.10 we present the results when filtering in the RGB and Luv color space. In general, filtering in Luv color space produces smoother images. This is due to two facts. The euclidean distance between two Luv values is perceptually meaningful, i.e. it is proportional to the distance of the colors as per-



(a) MF with Epanechnikov kernel (b) LMF with Epanechnikov kernel (c) CMS with Epanechnikov kernel (d) AD with Epanechnikov kernel



(e) MF with Normal kernel (f) LMF with Normal kernel (g) CMS with Normal kernel (h) AD with Normal kernel

(i) MF with Epanechnikov kernel (j) LMF with Epanechnikov kernel (k) CMS with Epanechnikov kernel (l) AD with Epanechnikov kernel

(m) MF with Normal kernel (n) LMF with Normal kernel (o) CMS with Normal kernel (p) AD with Normal kernel

Figure 2.7: Epanechnikov vs Normal kernel experiment. We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 19$. All the images are processed in RGB color space.

(a) MF with Epanechnikov kernel (b) LMF with Epanechnikov kernel (c) CMS with Epanechnikov kernel (d) AD with Epanechnikov kernel

(e) MF with Normal kernel (f) LMF with Normal kernel (g) CMS with Normal kernel (h) AD with Normal kernel

(i) MF with Epanechnikov kernel (j) LMF with Epanechnikov kernel (k) CMS with Epanechnikov kernel (l) AD with Epanechnikov kernel

(m) MF with Normal kernel (n) LMF with Normal kernel (o) CMS with Normal kernel (p) AD with Normal kernel

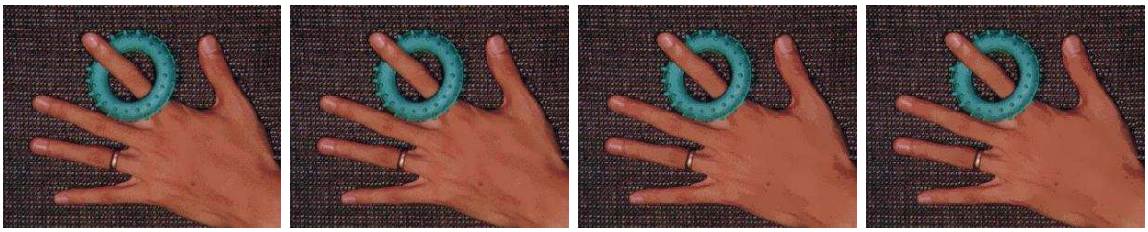
Figure 2.8: Epanechnikov vs Normal kernel experiment. We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 19$. All the images are processed in RGB color space.

ceived by a human observer. This is not true in RGB, where very similar colors might be located far away and vice versa. Furthermore the range of values for each component (L, u, v) is different (for example in our implementation $L \in [0 \dots 100]$, $u \in [-100 \dots 180]$, $v \in [-135 \dots 110]$.), while each of the Red, Green and Blue components have values from 0 to 255.

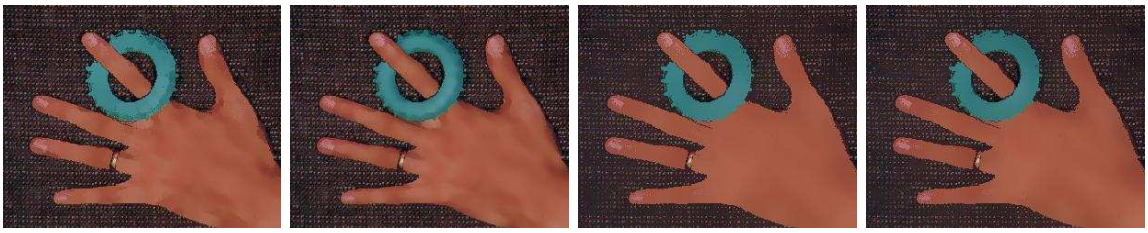
In these experiments, mode finding and local mode filtering seem to produce almost identical images, while color mean shift preserves the boundaries better than anisotropic diffusion. Both latter methods smooth the image considerably more than the former ones.

2.4.3 Color uniformity of regions after filtering

Next we compare the ability of the filtering algorithms to suppress texture and produce uniform regions. State of the art approaches to locate and classify texture use filter responses [32], [33] clustered in an K nearest neighbors framework. We measure, instead, the uniformity of the regions using zero (i.e. color histogram) and first order (i.e. gradient magnitude histogram) statistics on the color space. We compute the magnitude of the image gradient for each color channel on every image point using a 3×3 Sobel filter. In Figs. 2.11, 2.12, 2.13, 2.14 we display the histograms of the color and gradient distributions for the original images as well as the filtered ones with a Normal kernel in Luv color space. The difference between the filtering results is most obvious in the "hand" image. In color mean shift filtered image the vast majority of gradient magnitudes are close to zero. A comparable



(a) MF on RGB color space (b) LMF on RGB color space (c) CMS on RGB color space (d) AD on RGB color space



(e) MF on LUV color space (f) LMF on LUV color space (g) CMS on LUV color space (h) AD on LUV color space

(i) MF on RGB color space (j) LMF on RGB color space (k) CMS on RGB color space (l) AD on RGB color space

(m) MF on LUV color space (n) LMF on LUV color space (o) CMS on LUV color space (p) AD on LUV color space

Figure 2.9: RGB vs Luv color space experiments (1/2). We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 5$. All the images are processed with a Normal kernel.

(a) MF on RGB color space (b) LMF on RGB color space (c) CMS on RGB color space (d) AD on RGB color space

(e) MF on LUV color space (f) LMF on LUV color space (g) CMS on LUV color space (h) AD on LUV color space

(i) MF on RGB color space (j) LMF on RGB color space (k) CMS on RGB color space (l) AD on RGB color space

(m) MF on LUV color space (n) LMF on LUV color space (o) CMS on LUV color space (p) AD on LUV color space

Figure 2.10: RGB vs Luv color space experiments (2/2). We use $h_s = 5$ (resulting in a window of 11×11 pixels) and $h_r = 5$. All the images are processed with a Normal kernel.

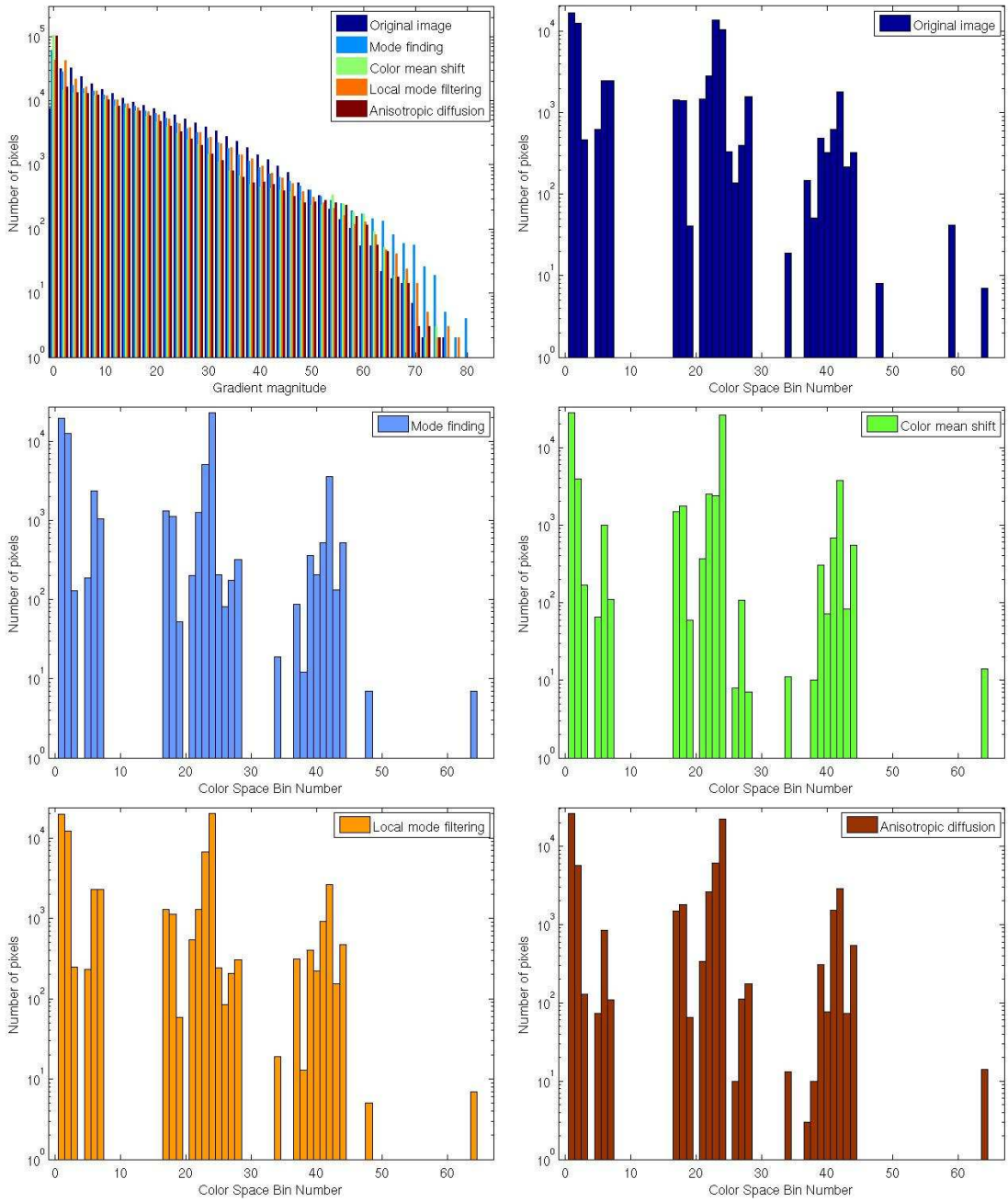


Figure 2.11: Histograms for the original hand image and the processed results of Fig. 2.9 second row. Notice that in all figures the Y axis is in logarithmic scale. The color mean shift filtered image uses the least number of color bins and exhibits less gradient variation compared to all the other methods and the original image.

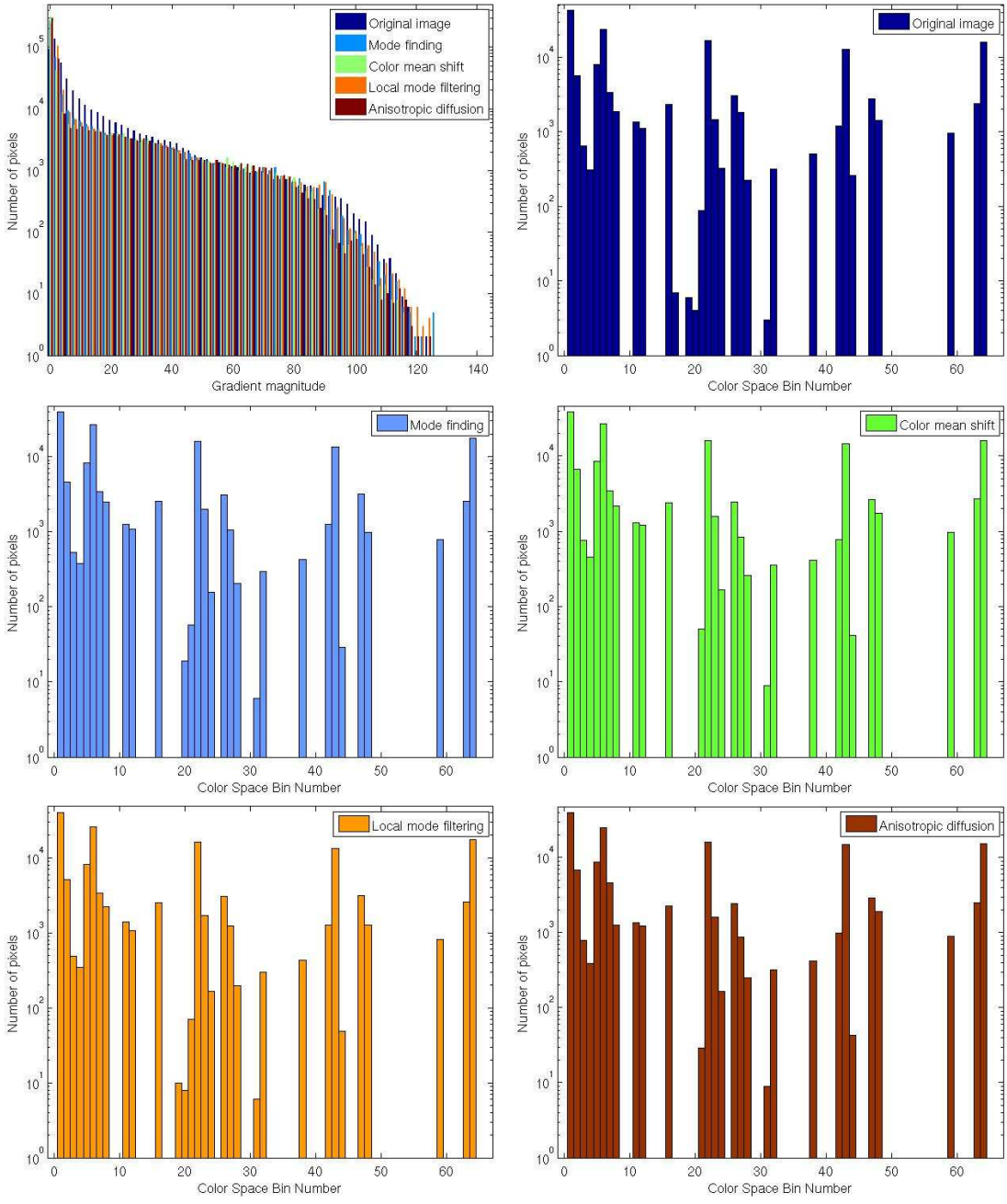


Figure 2.12: Histograms for the original workers image and the processed results of Fig. 2.9 fourth row. Notice that in all figures the Y axis is in logarithmic scale.

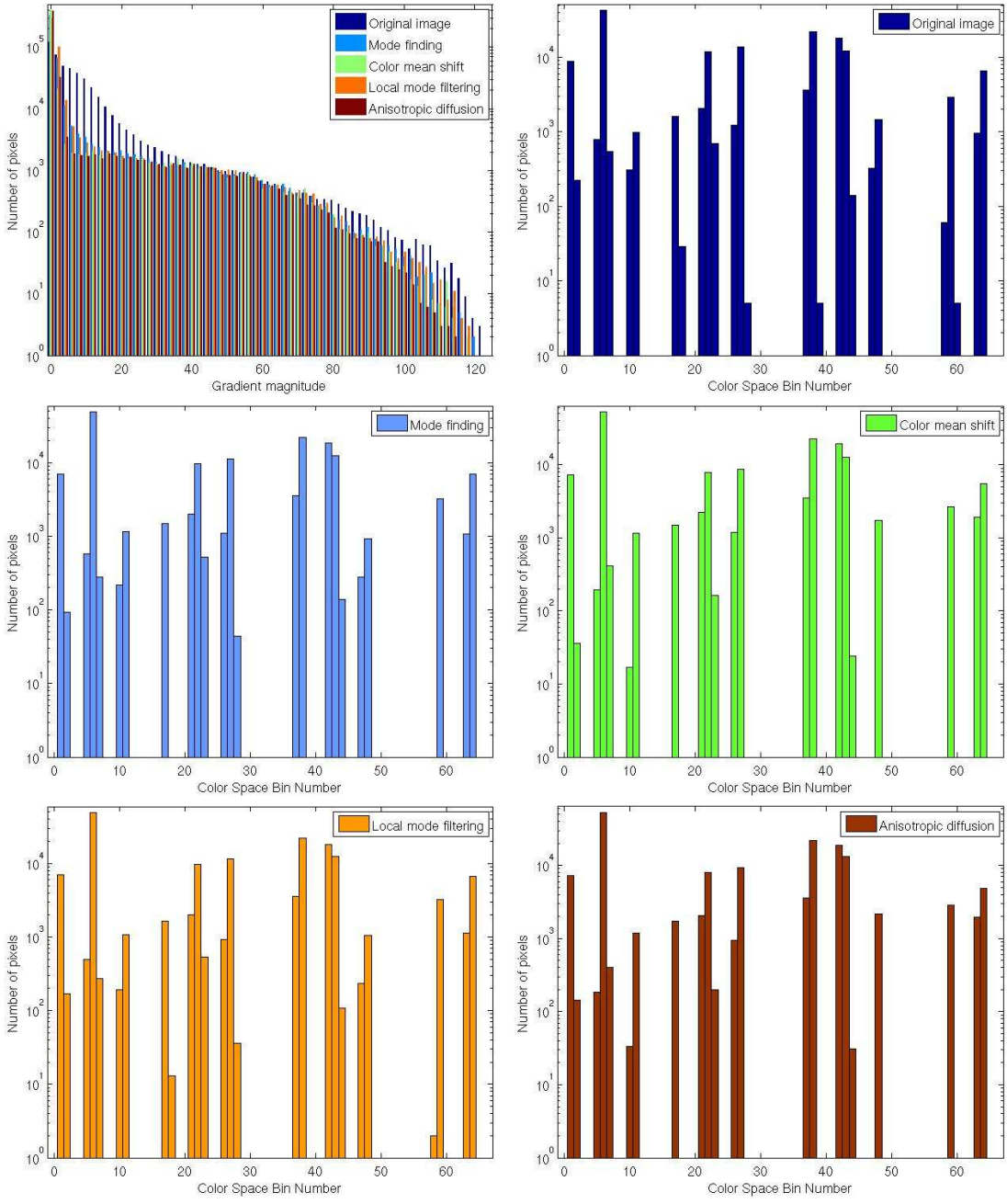


Figure 2.13: Histograms for the original woman image and the processed results of Fig. 2.10 second row. Notice that in all figures the Y axis is in logarithmic scale.

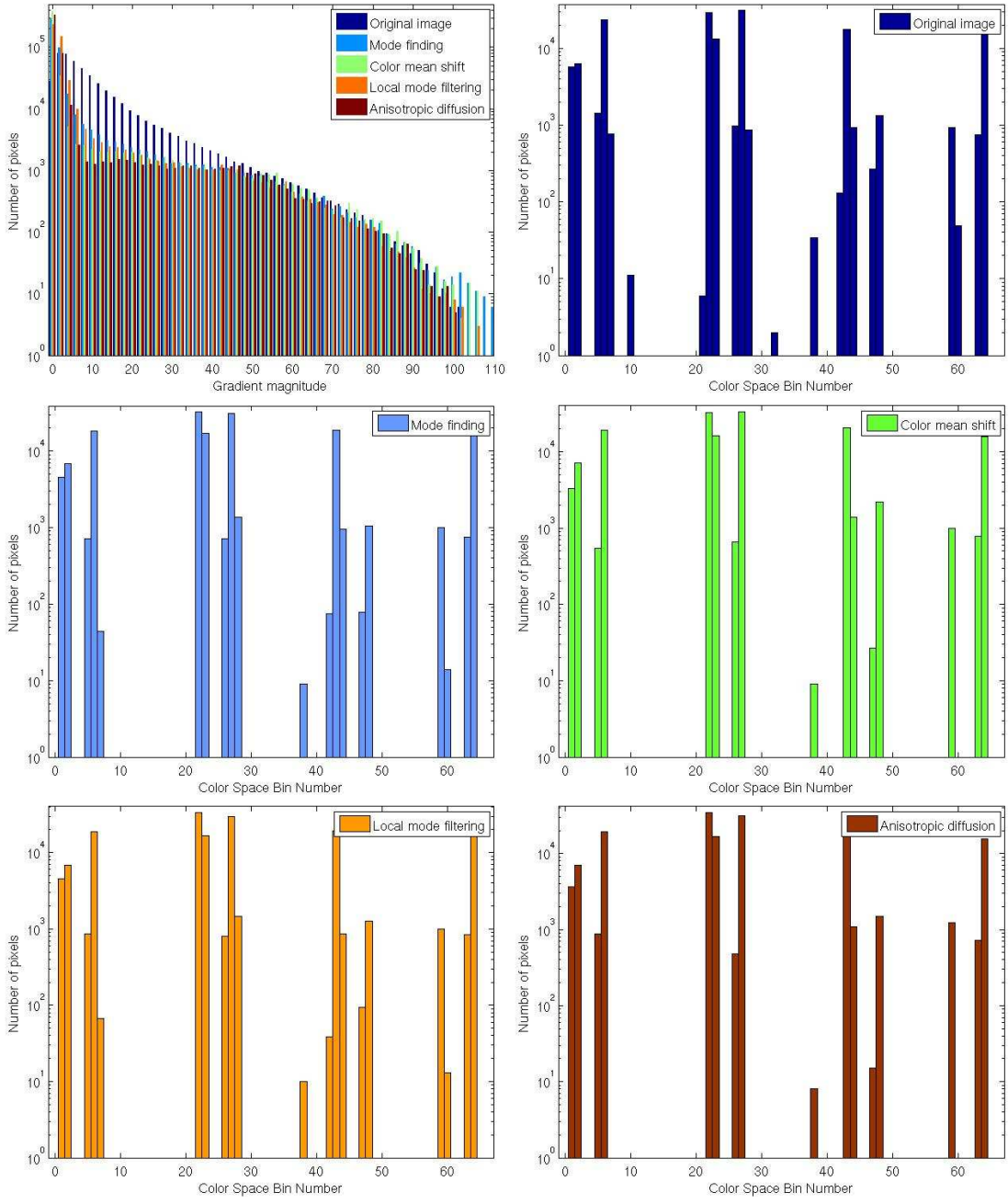


Figure 2.14: Histograms for the original houses image and the processed results of Fig. 2.10 fourth row. Notice that in all figures the Y axis is in logarithmic scale.

number of magnitudes are close to zero in anisotropic diffusion image as well. In mode finding and local mode filtered image half as many pixels and in the original image 3% as many pixels have gradient close to zero. In the same figures, we display the joint color histogram for the five images. As expected, in the color mean shift image the pixels are clustered to fewer color bins compared to the other images.

In Table 2.2 we display the entropy measure for the color distribution and the gradient magnitude for each method with the different kernels and color spaces (and constant spatial and color resolutions $h_s = 5$, $h_r = 5$). The entropy definition[4] measures how "random" an image is. Thus, an image created by sampling each pixel's color value from a uniform random distribution is expected to have a large entropy value, while a single uniform color image has an entropy of 0. In general lower entropy values indicate more uniform colored images, i.e. images with less number of segments of more uniform color. From the results of Table 2.2 one can reach the following conclusions.

- Color mean shift produces the least variation on the color and gradient histogram, followed by anisotropic diffusion, mode finding and local mode filtering.

- Within a filtering method the differences between the different kernels and color spaces are small for the color entropy measures but quite significant for the gradient measures. The least entropy measures for the gradient magnitude are obtained when we use Normal kernel and perform the processing in the

[4]If X is a discrete random variable with possible values $\{x_1, \ldots, x_n\}$ then the entropy is defined as $H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$, where b is the base of the logarithm (in our case we use $b = 2$).

Table 2.2: Entropy measures for the color and gradient histograms for the four images after performing the filtering with different methods and different kernels in the two color spaces. The first number is the entropy for the color and the second for the gradient histogram. The lower the values the smaller the variation.

Hand Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	**6.14**, 12.97	**6.14**, 12.97	**6.14**, 12.97	**6.14**, 12.97
Epanechnikov, Luv	7.02, 12.91	7.02, 12.91	7.42, 12.82	7.50, 12.83
Normal, RGB	7.15, 12.68	7.32, 12.59	8.91, 11.89	9.32, 11.94
Normal, Luv	10.47, 10.85	11.20, 11.02	9.84, **8.87**	10.93, 9.16

Workers Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	13.95, 9.59	14.64, 9.59	12.34, 9.21	13.31, 9.35
Epanechnikov, Luv	13.72, 8.78	14.70, 8.75	12.51, 8.16	13.59, 8.21
Normal, RGB	12.46, 8.47	14.16, 8.48	**10.82**, 7.85	12.61, 8.14
Normal, Luv	12.74, 7.05	14.31, 7.16	11.80, **6.17**	13.16, 6.28

Woman Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	14.25, 8.49	14.58, 8.43	13.12, 8.43	13.79, 8.39
Epanechnikov, Luv	13.67, 7.30	14.37, 7.15	12.37, 6.13	13.24, 6.07
Normal, RGB	13.26, 7.72	14.16, 7.41	**11.58**, 7.51	12.81, 7.35
Normal, Luv	13.08, 5.18	13.92, 5.11	12.07, **4.23**	12.86, 4.30

Houses Image	Mode finding	Local Mode filtering	Color Mean Shift	Anisotropic Diffusion
Epanechnikov, RGB	14.27, 9.12	14.59, 9.07	13.07, 9.04	13.70, 8.98
Epanechnikov, Luv	13.39, 7.75	14.17, 7.60	11.71, 6.29	12.78, 6.46
Normal, RGB	13.05, 8.53	14.10, 8.22	**10.94**, 8.08	12.53, 8.12
Normal, Luv	12.72, 5.57	13.62, 5.67	11.48, **4.36**	12.56, 4.71

Luv color space.

- When processed with the Epanechnikov kernel in the RGB color space all the methods produce very similar results. The difference between the methods is emphasized when the processing involves a Normal kernel and the Luv color space.

- In the case of the hand image the color resolution that we use ($h_r = 5$) is too small to eliminate the textured background and the color variation inside the hand (as it is shown in Fig. 2.9). That is why we obtain these results.

Overall these facts allow us to claim that color mean shift produces the most uniform regions, followed by anisotropic diffusion. Mode finding and local mode filtering produce very similar results. A natural question to ask is whether the above results are due to *over smoothing*. From the sample filtering results presented above this does not seem to be the case. The only way to verify that though is to perform the segmentation and then compare the results against human segmented images. In Sec. 3.4 we present these experiments. As we discuss there the segmentation results for color mean shift are better than the ones for the other filtering methods, thus we can safely conclude that *color mean shift produces more uniform regions without over smoothing the original image*.

2.4.4 Filtering speed comparison

An objective comparison of the filtering speed of the different methods is not a simple task. Besides the implementation details that greatly affect the speed, there

is also a number of algorithmic parameters that can significantly speedup or slow down the convergence of the optimization procedure. We start our comparison by evaluating the role of these parameters and then we discuss whether general speed up techniques that have been proposed in the literature can be applied to the different methods or not. For fairness sake, we use our own implementation of all the filtering methods that consists of Matlab files for the image handling and the general input/output interface, while the optimization code is written in C. We perform all the experiments on a desktop computer with an Intel Core2 Quad CPU @$3GHz$[5].

2.4.4.1 Image size

The number of pixels directly affect the filtering speed. In theory the complexity of the algorithm increases linearly with the number of pixels, since each pixel represents a feature vector that needs to be processed. The theoretical prediction is verified in practice as Fig. 2.15 shows.

2.4.4.2 Spatial resolution (h_s)

Theoretically, all the filtering methods (but Mean Shift and Spatial Mean Shift) depend quadratically on the spatial bandwidth. In practice, other parameters, explained below, make the dependence less than quadratic. Fig. 2.16 displays the filtering speed with respect to the spatial resolution for the methods, when all the other parameters are the same.

[5]Due to Matlab's limitation only one core is used in the experiments.

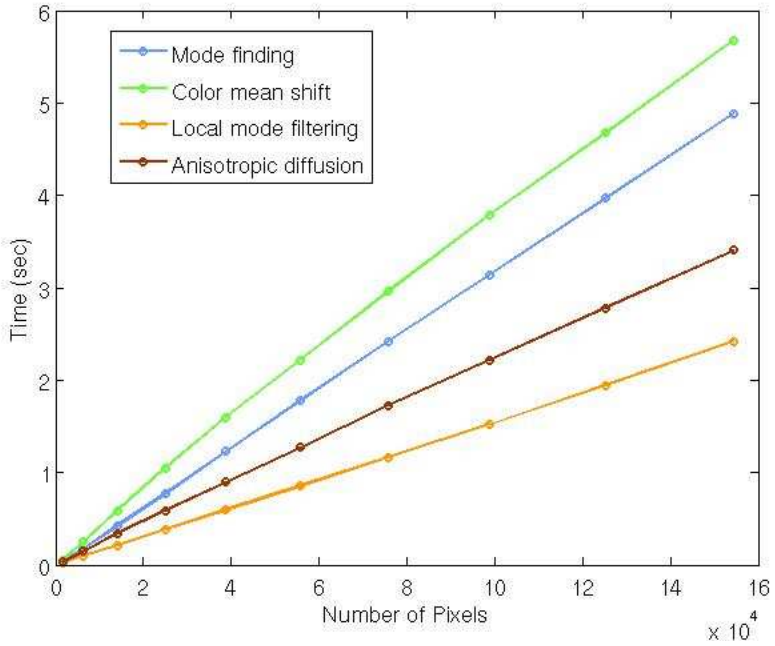


Figure 2.15: The filtering speed as a function of the image size (i.e., number of pixels) for all four methods. We use the "workers" image (whose original size is 321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel with spatial and color resolutions $h_s = 5, h_r = 15$ respectively. We also limit the number of iterations to 20 and the convergence threshold is 0.001. We perform the filtering 5 times for each image size and only plot the median value.

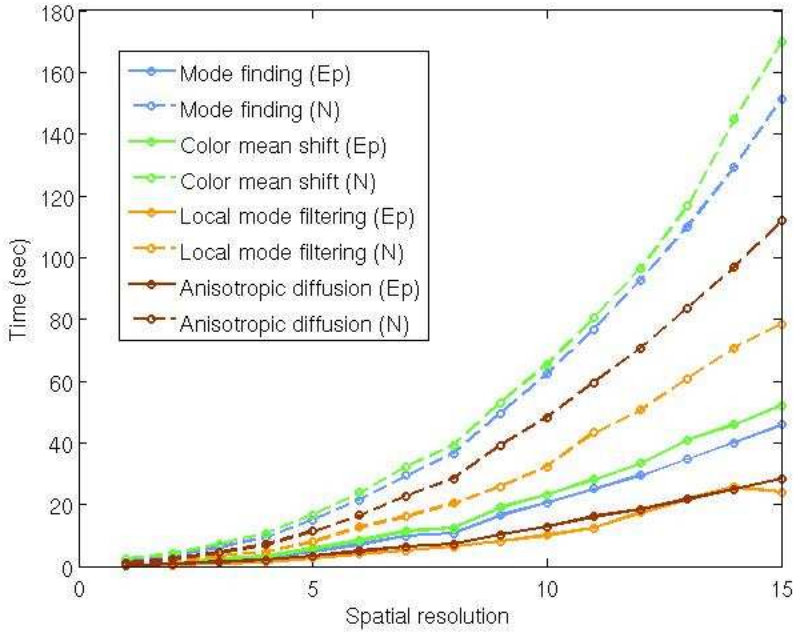


Figure 2.16: The filtering speed as a function of the spatial resolution (h_s) for all four methods. We use the "workers" image (321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel (continuous line) or Normal kernel (dotted line). We also limit the number of iterations to 20 and stop the optimization for pixels that move less than 0.001 between two iterations. We perform the filtering 5 times for each value of h_s and only plot the median value.

2.4.4.3 Epanechnikov vs Normal kernel

For each pair of pixels, computation of the weight using the Epanechnikov kernel only requires a comparison, while the calculation of an exponential number is necessary for the case of the Normal kernel. As a result the former operation is much cheaper than the latter and thus filtering with an Epanechnikov kernel is faster compared to filtering with a Normal kernel as is shown in Fig. 2.16. Other researchers (e.g. **[34]**) have proposed the use of lookup tables to approximately compute the exponents much faster.

At this point we should note that the overall speed of the segmentation process is also affected by the quality of the result of the filtering process. We experimentally found, that using a normal kernel produced better results and as a consequence sped up the grouping step. Overall the use of a Normal kernel still resulted in slower segmentation times, but the time difference was not as large as Fig. 2.16 shows.

2.4.4.4 Convergence threshold

As described above, on each iteration of the optimization procedure each pixel vector is compared against its neighbors and shifted. If this shift is less than a predefined value (denoted convergence threshold) then we ignore that pixel in subsequent iterations of the optimization procedure. Intuitively the convergence threshold denotes how close to the "true" solution the optimization should reach before termination. At this point we would like to emphasize that for the mode finding and the local mode filtering methods the shift of each pixel is a monotonically decreasing function

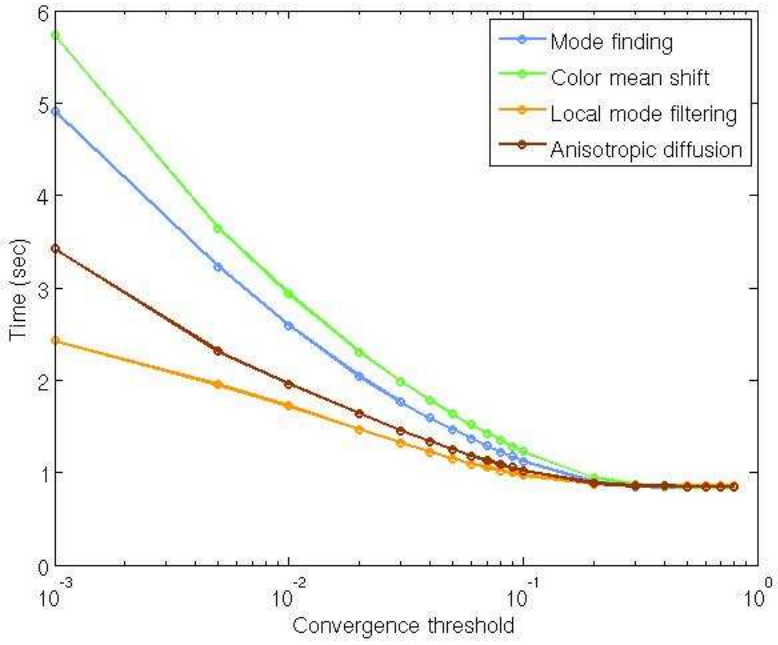


Figure 2.17: The filtering speed as a function of the convergence threshold for all four methods. We use the "workers" image (321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel with spatial and color resolution $h_s = 5, h_r = 15$ respectively. We also limit the number of iterations to 50. We perform the filtering 5 times for each value of the convergence threshold and only plot the median value. Notice that the X-axis is on logarithmic scale.

of the iteration number, while for color mean shift and anisotropic diffusion it is not. Fig. 2.17 displays the filtering speed with respect to the convergence threshold. As expected the higher the threshold the faster the filtering. Especially for thresholds less than 0.1 the filtering time decreases almost exponentially. According to this graph and all the previous ones, local mode filtering is the fastest filtering operation followed by anisotropic diffusion, and then mode finding, while color mean shift is slightly slower. This is expected due to the extra number of calculations needed to estimate the $5D$ feature vector instead of the $3D$ feature vector in the other methods.

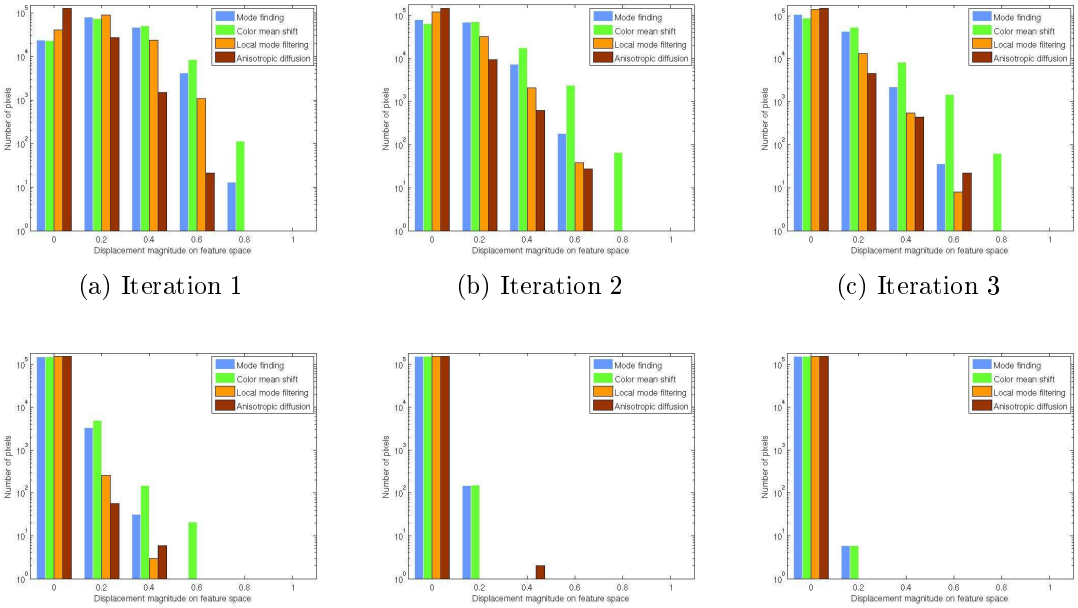
2.4.4.5 Feature vector displacement per iteration

Related to the previous parameter, here we evaluate the convergence speed of the filtering algorithms, namely how many iterations are requited for all the pixels to reach the convergence threshold. In Fig. 2.18 we plot the histogram of the displacement of the feature vectors on a single iteration. Although it is hard to make any definite conclusions one observes that in the first three iterations color mean shift displaces pixels more than any other method. Overall, local mode filtering and anisotropic diffusion converge (i.e. all the pixels are displaced less than 0.2) in 17, 20 iterations respectively. Mode finding and color mean shift converge much slower requiring 40 and 39 iterations respectively. Similar behavior was observed in all the examples that we used for testing. This leads us to believe that color mean shift converges as least as fast as mode finding.

2.4.4.6 Filtering speed conclusions

As we said before we use our own implementation of all the filtering methods, that is a straightforward translation of Table 2.1 to Matlab and C code, to perform the speed experiments. A number of methods can be used to perform the filtering faster.

In the core of all the filtering algorithms the pairwise distance between feature points needs to be computed for all pairs of points. As suggested in [11] employing data structures and algorithms for multidimensional range searching can speed up the filtering. This technique can be used in all the filtering methods and is expected to significantly improve the speed of slow methods such as mean shift and spatial



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

(d) Iteration 10 (e) Iteration 20 (f) Iteration 30

Figure 2.18: The histograms of vector displacements for a number of iterations for all four filtering methods. We use the "workers" image (321×481 pixels) and perform the filtering on the RGB color space with an Epanechnikov kernel with spatial and color resolutions $h_s = 5, h_r = 15$ respectively. We also limit the number of iterations to 40.

mean shift.

In mode finding the trajectory of most feature points lay along the path of other feature points. Christoudias et al. in [35] report a speed up of about five times relative to the original algorithm when they "merge" the feature points together. This trick can directly be used in local mode filtering. A variation of the same concept could also be used to speed up the filtering in all the other methods.

Paris and Durant in [5] suggest a fast method to find the local modes of the $5-D$ features points coming from large color images. Contrary to the title of their work their method is based on directly estimating the kernel density on a sparse $5-D$ grid. Even though this idea is appealing and alleviates the computational problem associated with increasing the spatial kernel resolution h_s, it is not clear how it can be used to speed up any of the filtering methods.

In the same paper ([5]) extra computational reduction is achieved by reducing the dimensionality of the feature space from $5-D$ to $4-D$ (or $3-D$). Principal component analysis is used to perform the reduction and the authors report that a reduction to $4-D$ results in almost no loss of filtering quality, while the filtering is performed 5 times faster. This is to be expected for their method, since they sample the whole feature space. The algorithms that we study, though, would benefit little (if at all) from such a technique since the additional cost of performing the PCA would offset the gain of performing the filtering in less dimensions.

The introduction of the multicore CPUs and, especially, GPUs has provided new way to improve the execution speed of algorithms through a parallel implementation. From Table 2.1 and Fig. 2.4 it is clear that the filtering of each feature point

can be performed in parallel. We expect that a careful implementation of any of the four algorithms (i.e. mode finding, color mean shift, local mode filtering and anisotropic diffusion) on a modern GPU will run in real time for VGA or larger images.

2.5 Conclusions

In this chapter we presented a unifying framework under which we can express different filtering algorithms. Using the new understanding of filtering, we developed three new edge preserving filtering methods, that we named Color Mean Shift, Spatial Mean Shift and Joined Bilateral Filtering. The first one exhibits similar clustering characteristics with the original Mean Shift method while being almost as computationally efficient as the Mode Finding method, so it was included in our filtering comparison. We performed a comparison of four different methods (Mode Finding, Color Mean Shift, Local Mode Filtering and Anisotropic diffusion) on a number of images with different configurations for the color space and the kernel function. Overall we noticed that Color Mean Shift outperforms (i.e. creates more uniform segments with better boundary separation) than the other methods with the drawback of being slightly slower. Table 2.3 synopsizes the results of the experimental comparison for performing edge preserving filtering.

Table 2.3: Synopsis of the filtering results

- Normal kernel gives smoother filtering results compared to Epanechnikov kernel
- Luv color space produces smoother filtering results compared to RGB color space.
- Mode finding and local mode finding produce similar filtering results. Mode finding performs slightly better filtering.
- Color mean shift and anisotropic diffusion produce similar filtering results. Color mean shift preserves the edges better than anisotropic diffusion.
- $3-D$ filtering (i.e. local mode filtering) is almost equivalent to $5-D$ filtering (i.e. mode finding) when the original image is used for the comparison. When the image obtained in the previous iteration is used then $5-D$ filtering (i.e. color mean shift) preserves edges better than $3-D$ filtering (i.e. anisotropic diffusion).
- Whether we use the original image for comparison or not affects the filtering more than whether we perform it in $3-D$ or $5-D$.
- Local mode filtering is the fastest; mode finding and local mode filtering are a little bit slower; color mean shift is even slower. All the methods are fast enough to perform the filtering in real time for a reasonably large image when implemented in GPUs.

Chapter 3

Color Based Segmentation as a Two Stage Process

3.1 Introduction

The edge preserving filtering framework, that we presented in the previous chapter, is the first component of a color-based segmentation system. In this chapter we present the other component, namely clustering algorithms for pixels (or feature points) on 3D (or 5D) space. First, we briefly describe the grouping algorithms that we use in the segmentation experiments; a greedy connected components method with a fixed threshold, its variant using Region Adjacency Graph [35] and its extension using an adaptive threshold [7].

Then, we experimentally compare all the combinations of filtering and grouping techniques using the Berkeley dataset [36]. In our comparison we focus on three criteria; correctness, robustness with respect to the parameters and robustness with respect to image selection. We use both boundary and region based measures for comparison. More specifically, we consider the percentage of edges retrieved and the edge distance between segmentations as the boundary based criteria. We also compute the Global Consistency Error[36], the Rand Index[37] and the Variation of Information [38],[39] region based measures.

3.2 Grouping methods

A variety of grouping methods exist in the literature for image segmentation. As a matter of fact almost all the color based image segmentation methods are grouping methods. Next, we describe the three methods that we have chosen to use in the segmentation experiments. The first method is a simple connected components algorithm with a global threshold, while the other two methods are extensions of that algorithm. All methods are simple, namely they don't require the use of complicated tuning parameters and they are used widely for image segmentation. Another advantage is that they are fast so they can be used for (almost) real time segmentation.

3.2.1 Greedy Connected Components grouping (CC3D and CC5D)

This is the same strategy that Comaniciu and Meer implicitly use in their image segmentation algorithm [11]. The method is a good starting point for our comparison; its simplicity allows us to compare the smoothing algorithms for the task of segmentation without worrying that the result has been "changed" by the grouping algorithm. Thus, the quality of the segmentation is directly related to the quality of the filtering.

In a nutshell, the algorithm groups neighboring pixels together if and only if their Euclidean distance is within a user defined threshold. Note that there is a $3-D$ and a $5-D$ variant of this algorithm since pixel x_i is represented by either a $3-D$ vector ($\mathbf{S}_i$) or a $5-D$ vector ($[\mathbf{x}_i, \mathbf{S}_i]$) (Fig. 3.1). In our implementation we

use an union-find data structure to perform the merging so the complexity of the algorithm is almost linear on the number of pixels.

The biggest problem with this simple grouping method is the "segment diffusion" problem, when two quite different segments are merged together because there is a single weak (blurry) edge between them (e.g. the clouds and the sky are merged into a single segment in the first images of the top row of Fig. 3.2). In order to reduce the impact of this problem we reduce the grouping threshold (t in Fig. 3.1, top row) to 0.5.

3.2.2 Grouping using Region Adjacency Graphs (GRAG)

This is the grouping method proposed in [35] and used in the EDISON segmentation system. Conceptually this method is similar to the connected components method (i.e. a hard threshold of $t = h_r/2$ is used), but the use of region adjacency graphs produces slightly different segmentation results. We should note that the above methods are invariant to the merging order of the pixels.

3.2.3 Grouping with an Adaptive Threshold (GAT)

Felzenszwalb and Huttenlocher in [7] present a variation of the connected component algorithm where an adaptive threshold for merging segments is used. Each segment C_i keeps track of the maximum distance between two pixels belonging to it[1](denoted $Int(C_i)$) and two segments $C_i,\ C_j$ are merged only if the minimum distance between the pixels belonging to their common boundary is smaller than the internal distance

[1]Only the edges belonging to the minimum spanning tree of the segment are considered

$Int(C_i)$, $Int(C_j)$. The method is described in Fig. 3.1. This algorithm is also linear on the number of pixels.

In the experiments, unless otherwise noted, we use the values of 0.5, 500 for σ, k respectively for the grouping parameters. This are the values suggested by the authors in [7].

3.3 Segmentation as filtering+grouping

The notion of segmentation consisting of a filtering followed by a grouping step is not new, but it is underemphasized in the literature. Most image segmentation (i.e. grouping) algorithms operate on the original image, while the filtering algorithms are usually applied to the problems of edge preserving smoothing or noise removal. Comaniciu and Meer [11] talk about segmentation consisting of a filtering and a fusion step, but they focus on the filtering step and they use the simple connected component algorithm of Fig. 3.1 top left, to obtain the final segments. Subsequent work from the same group [35] focuses on how to bring edge information into the filtering and grouping step, but they still use a similar connected components algorithm. Close to our philosophy is the work of Unnikrisnan et al. [40] where they combine the filtering algorithm of [35] with the grouping algorithm of [7]. Their focus, thought, is to introduce a new measure called Normalized Probabilistic Rand to compare the quality of segmentation.

One of the main points of this chapter is that both steps are important to obtain good segmentation results. In Fig. 3.2, for example, we present the seg-

Connected Components $3D$ (CC3D)

Input:
set of pixels $\mathbf{x}_i$ with intensities $\mathbf{S}_i$
a grouping threshold t
Output:
a set of labels (label l_i for $\mathbf{x}_i$)
Algorithm:
for all pixels $\mathbf{x}_i$
  assign label l_i
repeat until convergence
  for all pixels $\mathbf{x}_i$
    for all pixels x_j
      if $||\mathbf{S}_i - \mathbf{S}_j|| < t$ and
      $\mathbf{x}_i, \mathbf{x}_j$ have different labels
        merge the labels of
        $\mathbf{x}_i$ and $\mathbf{x}_j$ ($l_i \equiv l_j$)

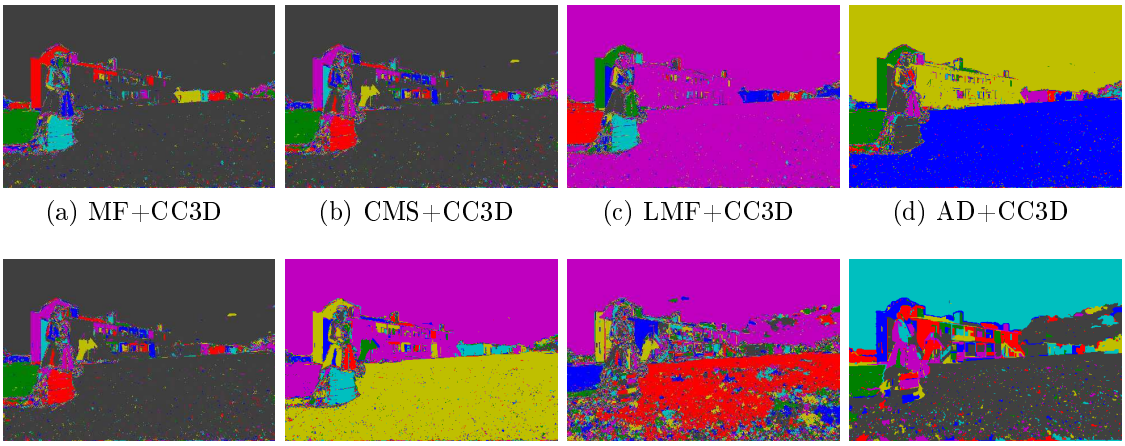
Connected Components $5D$ (CC5D)

Input:
set of pixels $\mathbf{x}_i$ with intensities $\mathbf{S}_i$
a grouping threshold t
Output:
a set of labels (label l_i for $\mathbf{x}_i$)
Algorithm:
for all pixels $\mathbf{x}_i$
  assign label l_i
repeat until convergence
  for all pixels $\mathbf{x}_i$
    for all pixels x_j
      if $||[\mathbf{x}_i, \mathbf{S}_i] - [\mathbf{x}_j, \mathbf{S}_j]|| < t$ and
      $\mathbf{x}_i, \mathbf{x}_j$ have different labels
        merge the labels of
        $\mathbf{x}_i$ and $\mathbf{x}_j$ ($l_i \equiv l_j$)

Grouping with an Adaptive Threshold (GAT)

Input:
An image as a graph $G = (V, E)$ with n vertices and m edges
Output:
A segmentation of V into components $S = (C_1, ...C_r)$
Algorithm:
sort E into $\pi = (o_1, \ldots, o_m)$ by non decreasing edge weight
in the initial segmentation S^0 each vertex v_i is its own segment
for $q = 1, \ldots, m$ construct S^q given S^{q-1} as follows
  let v_i, v_j be the vertices connected by the q^{th} edge $o_q = (v_i, v_j)$
  let pixels v_i, v_j belong to components C_i, C_j with
  $|C_i|, |C_j|$ number of elements respectively
  let $Int(C_i), Int(C_j)$ be the maximum edge weights of the
  minimum spanning tree of components C_i, C_j respectively
  let e_q be the weight of edge o_q
  if v_i, v_j belong to different components C_i, C_j and
  $e_q < \min\{Int(C_i) + \frac{k}{|C_i|}, Int(C_j) + \frac{k}{|C_j|}\}$
    merge C_i, C_j
return $S = S^m$

Figure 3.1: The grouping algorithms that we use in the segmentation experiments.



(a) MF+CC3D (b) CMS+CC3D (c) LMF+CC3D (d) AD+CC3D

(e) CMS+CC3D (f) CMS+CC5D (g) CMS+GRAG (h) CMS+GAT

Figure 3.2: On the first row we present the segmentation results when we use the same grouping method (CC3D) coupled with different filtering methods. On the second row we present the segmentation results when we use the same filtering method (Color mean shift) followed by a different grouping method.In the images each segment is represented by a different color. The filtering is performed on the RGB color space with an Epanechnikov kernel with spatial and color resolution $h_s = 5, h_r = 4$ respectively.

mentation results we obtained using different combinations of filtering and grouping methods. On the top row we use the same grouping method, namely CC3D, along with the four different grouping algorithms. It is clear that depending on the filtering method the sky is merged with the grass or not. On the second row the filtering method is kept constant (color mean shift) while the grouping method changes. Here the results significantly depend on the method, with the adaptive threshold method producing the most intuitive segments. In the next section we experimentally study the problem of color based segmentation by comparing different combinations of filtering and grouping algorithms. More specifically we couple each of the four filtering algorithms that we studied above with the four grouping algorithms that we introduced in the previous section to obtain a new segmentation method.

3.4 Segmentation Comparison

There is little effort to classify image segmentation algorithms and compare their characteristics due to two main factors. The multiplicity of methods each having a number of parameters make the comparison extremely tedious. Moreover, the "right" segmentation is hard to define, since there are many levels of detail in an image and therefore multiple different meaningful segmentations. S. Paris [5] for example, creates a hierarchical structure of segmentations where starting from a large number of segments, regions are merged together to create more coarse segmentations. Furthermore, in complex scenes the evaluation of a given segmentation mostly relies on subjective criteria. Borra and Shankar [3], for example, go as far as suggesting that the proper segmentation is task and domain specific. The difficulty of formally defining the quality of a segmentation explains the lack of segmentation databases for natural images.

The most complete attempt at comparing segmentation algorithms is presented on the Berkeley database and segmentation website [36]. Here a large set of images along with human created segmentations were made available for segmentation evaluation. This is the testbed we use in this chapter for the evaluation of the different segmentation methods[2]. More specifically we use the 200 training images along with the 1087 human created segmentations. Next, we first describe the different measures that we use for the comparison, and then we present the segmentation results.

[2]In Appendix A we also present segmentation results using the Weizmann Institute dataset [8].

3.4.1 Comparison measures

A number of measures have been proposed in the literature in order to compare two different segmentations of the same image. In general the segmentation measures can be classified in two categories; region based or boundary based. The first group includes measures that consider the overlap of the segments in the two segmentations, while in second consists of measures that count the overlap or the distance of the boundaries. From the measures that we use, the Global Consistency Error [36], the Variation of Information [38],[39] and the Probabilistic Rand index [37] are region based; Edge Percentage and Boundary Displacement Error [41] are boundary based.

Edge Percentage (EP) This is the simplest measure. We count the number of segmentation boundaries that coincide with the human annotated edges and divide by the total number of edges. In simple terms we compute the percentage of edges that the automatic segmentation is able to detect. In order to reduce the edge displacement problem we smooth both the computer generated boundary map and the human edge map with a small Normal kernel (3×3 in the experiments) and compute the sum of the piecewise dot product between the two maps[3]. This measure is not symmetric. Obviously the higher the value the more similar the two segmentations are.

Boundary Displacement Error (BDE) This quantity measures the average displacement error of the boundary pixels between two segmented images. Particularly,

[3]As a result the measure is not the edge percentage, so the Y-axis of the graphs should not be interpreted as such. Only the relative value for the two methods should be considered.

it defines the error of one boundary pixel in one segmentation as the distance between the pixel and the closest pixel in the other segmentation. BDE is not symmetric, thus we use it to measure the average distance of the human segmentation to the computer generated one. Intuitively, the lower the BDE value the more similar the two segmentations are. A BDE measure of 0 indicates that all the boundaries of the human segmentation are covered by the boundaries of the computer one, but not vice versa.

Global Consistency Error (GCE) This measure calculates the extent to which one segmentation can be viewed as a refinement of the other. Segmentations which are related in this manner are considered to be consistent, since they could represent the same natural image segmented at different scales. More specifically, a local error measure for each pixel is defined as the cardinality of the set difference between the two segments the pixel belongs to on the two segmentations, divided by the segment size. Then, the Global Consistency Error is defined as the average local error measure. This measure is symmetric and the lower the value the more similar the two segmentations. The two extreme segmentation cases, namely each pixel belonging to a separate segment and the whole image being a single segment both produce a zero value GCE. Thus, this measure is only suited for comparison of segmentations with approximately the same number of segments. In general the GCE range is $[0 \ldots 1]$.

Variation of Information (VI) This is an information theoretic criterion for comparing two groupings of the same data set. VI measures the amount of information

lost and gained in changing from the first to the second clustering. VI is positive, symmetric and obeys the triangle inequality (thus it is a metric on the space of groupings). Briefly, VI defines the distance between two segmentations as the average conditional entropy of one segmentation given the other, and thus roughly measures the amount of randomness in one segmentation which cannot be explained by the other. Being a distance metric the minimum value of VI is 0 while the maximum depends on the image size. The lower the value of VI the better the match between the two segmentations.

Probabilistic Rand Index (PR) This measure counts the fraction of pairs of pixels whose labellings are consistent between the computed segmentation and the ground truth, averaging across multiple ground truth segmentations to account for scale variation in human perception. PR is a measure of similarity and as such a value of 0 indicates no similarity, while a value of 1 indicates the highest similarity.

3.4.2 Results for varying color resolution h_r

To produce the first set of segmentation figures we only vary the value of the color resolution h_r of the filtering methods. More specifically, we let h_r to obtain values from 0.6 to 20 on increments of 0.3. We keep the remaining filtering parameters constant i.e., the maximum number of iterations for convergence is set to 20 and the convergence threshold to 0.1. We also use a spatial resolution of $h_s = 5$, resulting on a 11×11 smoothing window around each pixel. Furthermore, we utilize constant

parameters for the grouping methods. More specifically the grouping threshold (parameter t of Fig. 2.4) is set to 1 and 0.5 for the CC5D and CC3D grouping algorithms respectively. In the case of GRAG we use the fusion function of the EDISON toolbox provided by Christoudias et al.[35]. We use the excellent C++ code provided by Felzenszwalb and Huttenlocher [7] with parameters $\sigma = 0.5$ and $k = 500$ as suggested in their paper to implement the grouping with the adaptive threshold (GAD). In all the grouping methods the minimum number of pixels per region in set to 1.

We computed the comparison measures for each image of the database and further aggregated the results for the whole database using the median value[4]. These values are plotted on the Y-axis of each figure. On the X-axis we plot the average segment size, instead of the color resolution h_r. Thus all the plots below show the implicit curve of one comparison measure with respect to the average segment size. The motivation behind this choice is the following; a major goal of a segmentation algorithm is to create as large segments as possible without merging areas belonging to different objects. Some of the measures above (i.e. Edge Percentage, Boundary Displacement Error and Global Consistency Error) produce degenerate (and perfect) results when each pixel belongs to its own segment. Thus only those measures in conjunction with the segment size indicate whether a segmentation is good and useful or not. For the computation of the Boundary Displacement Error, the Global Consistency Error, the Variation of Information and the Probabilistic Rand Index

[4]Since the comparison measures vary significantly for different images we choose the median value as opposed to the mean value because it is more robust to outliers.

we use the code provided by J. Wright and A. Yang [42].

In the filtering experiments (Sec. 2.4) we observed that the selection of the color space and the filtering kernel greatly affects the amount of smoothing performed for a given color resolution. Hence, for the segmentation experiments we choose to perform the filtering over an extended range of color resolution. As a result, depending on the color space and kernel function, different ranges of color resolutions lead to oversegmentations and undersegmentations. We want to compare the "reasonable" segmentations, thus in the figures below we limit the maximum average segment size to 200, 500 or 1000 pixels (depending on the color space and kernel function used). Values above the corresponding threshold in each case clearly indicate a heavily undersegmented image (i.e. consisting of too few segments), as the value of all the measures verify.

In the previous sections we presented 4 different grouping methods and 4 different filtering methods. Considering that filtering can be performed in either RGB or Luv color space with Epanechnikov or Normal kernel, the total number of combinations is $2 \times 2 \times 4 \times 4 = 64$. Since, presenting the results of all 64 variations in a single graph would result in illegible figures, initially we group together the results for a specific selection of color space and kernel function and present these results on a single figure. Moreover, we produce a single graph for each of the 5 measures for a total of 20 figures.

While dividing the total number of curves by 4 simplifies the display, still plotting 16 curves on the same figure is hard. Instead of introducing a different color for each curve we follow the color convention of the filtering graphs. The

Table 3.1: Color convention for the segmentation plots

Color of	Line	Circle
Blue	Mode Finding (MF)	Connected Components in 3D (CC3D)
Green	Color Mean Shift (CMS)	Connected Components in 5D (CC5D)
Orange	Local Mode Finding (LMF)	Grouping using Region Adjacency Graphs (GRAG)
Brown	Anisotropic Diffusion (AD)	Grouping with an Adaptive Threshold (GAT)

color of the line indicates the filtering method, while the color inside the point circles indicates the grouping method that is used. Table 3.1 displays all the color combinations.

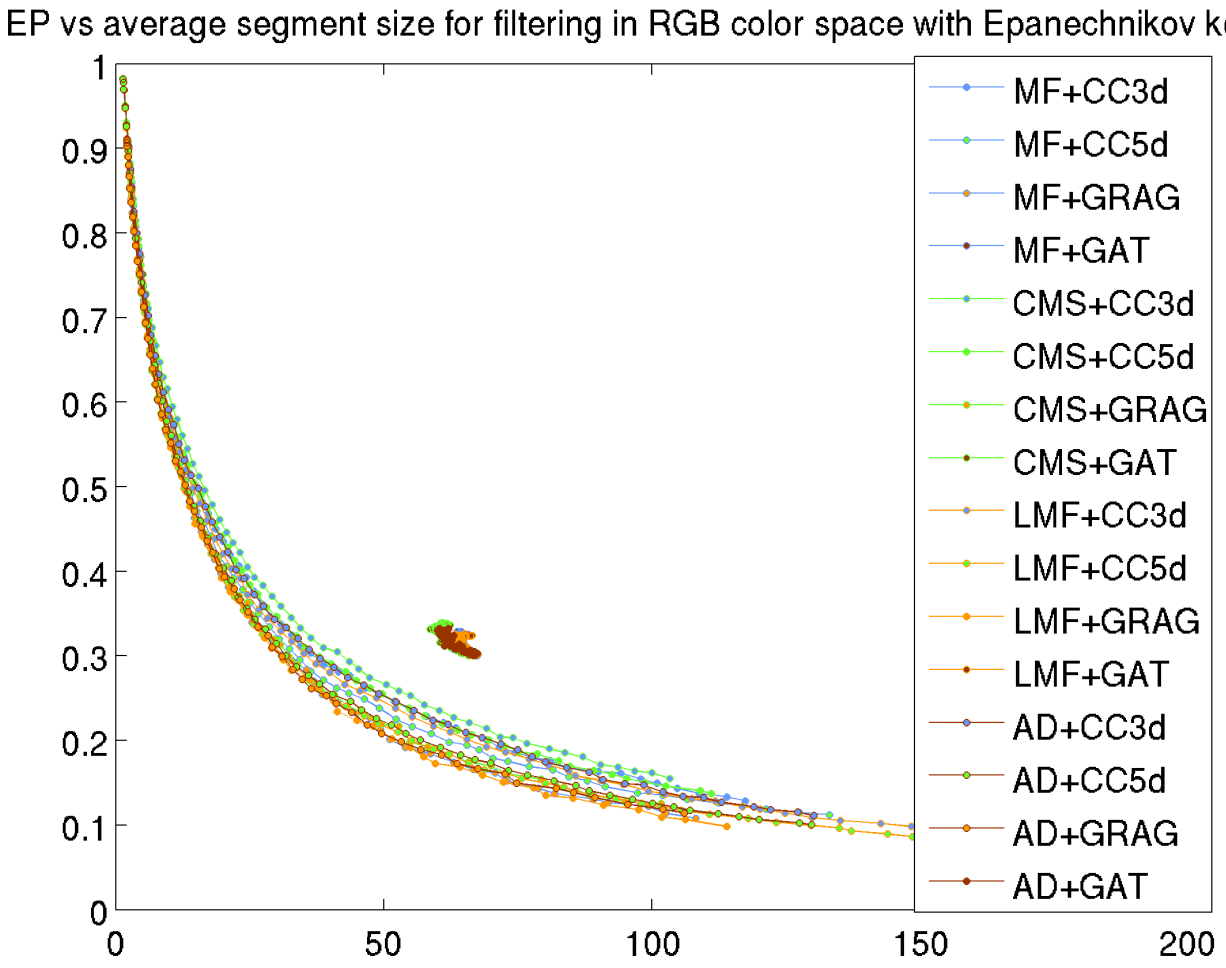


Figure 3.3: Edge Percentage vs average segment size plots when filtering is performed in the RGB color space with an Epanechnikov kernel.

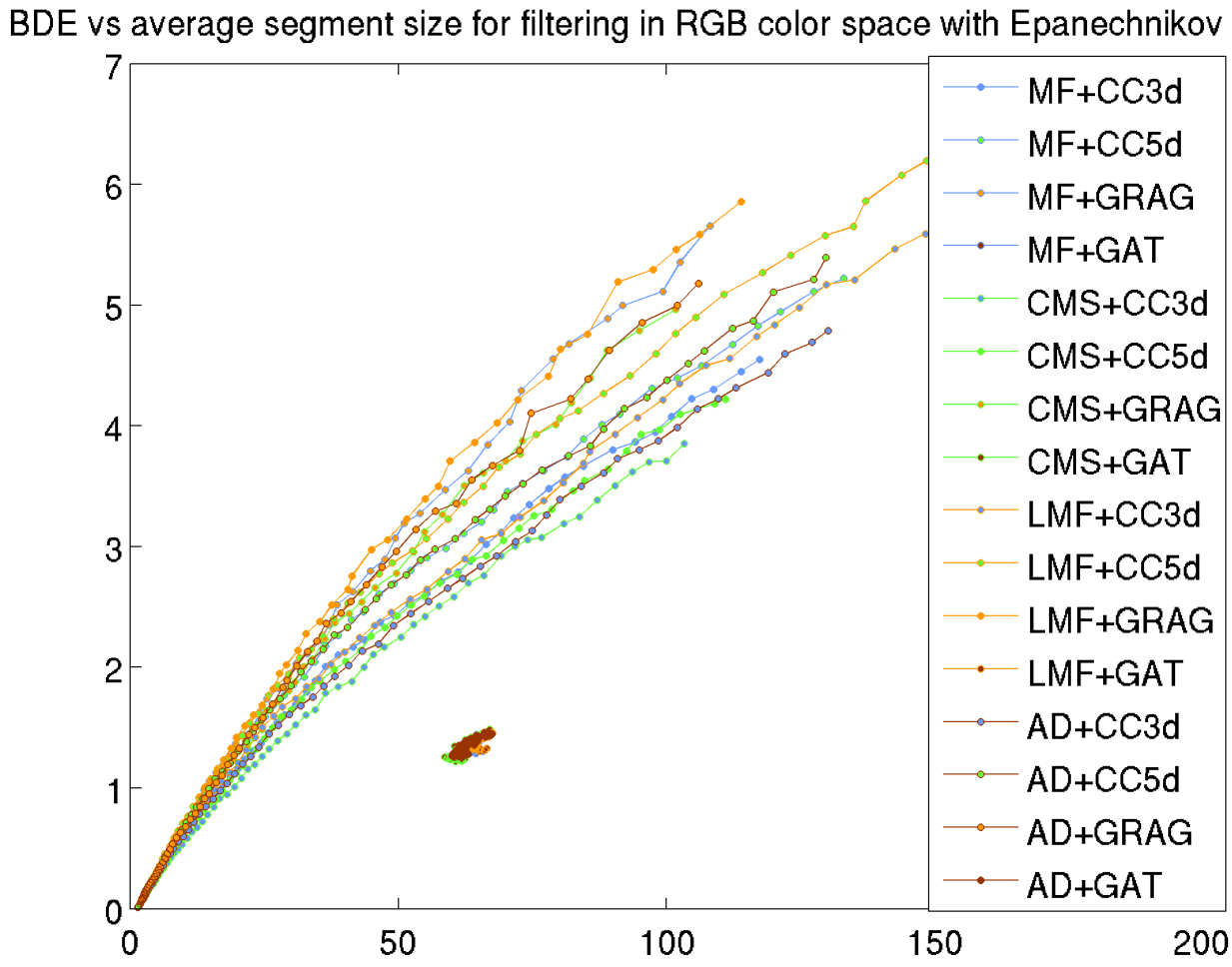


Figure 3.4: Boundary Displacement Error vs average segment size plots when filtering is performed in the RGB color space with an Epanechnikov kernel.

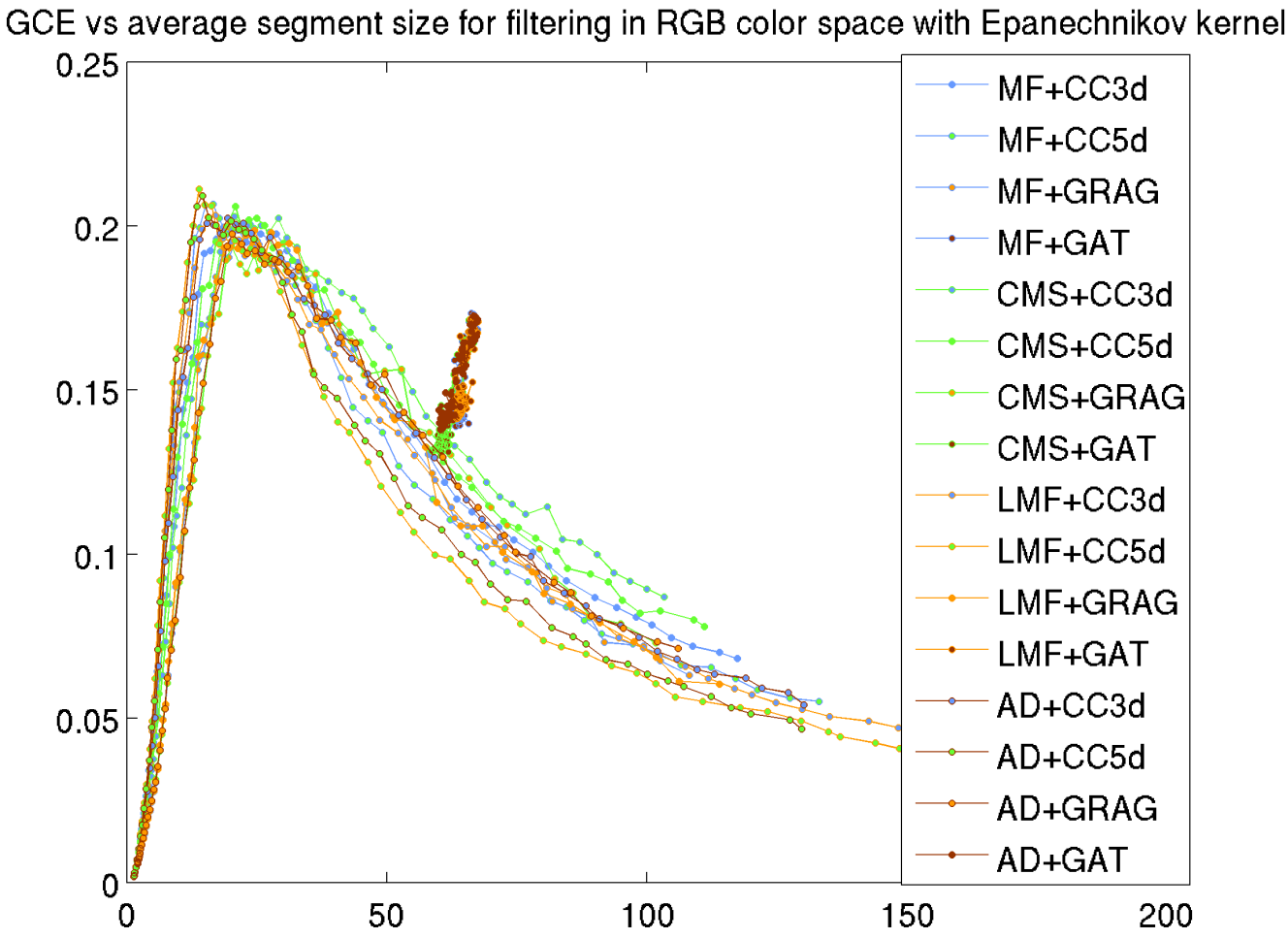


Figure 3.5: Global Consistency Error vs average segment size plots when filtering is performed in the RGB color space with an Epanechnikov kernel.

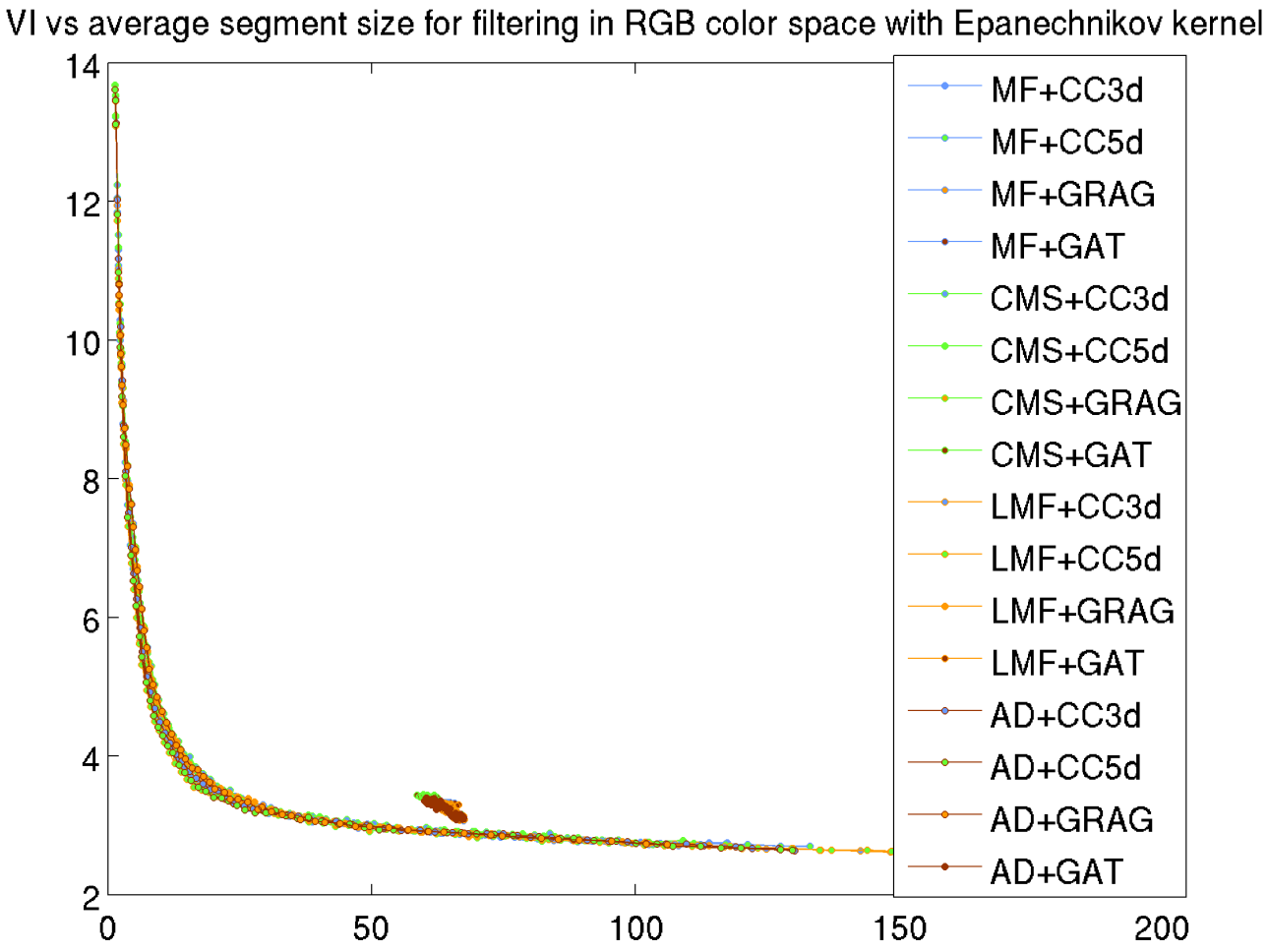


Figure 3.6: Variation of Information vs average segment size plots when filtering is performed in the RGB color space with an Epanechnikov kernel.

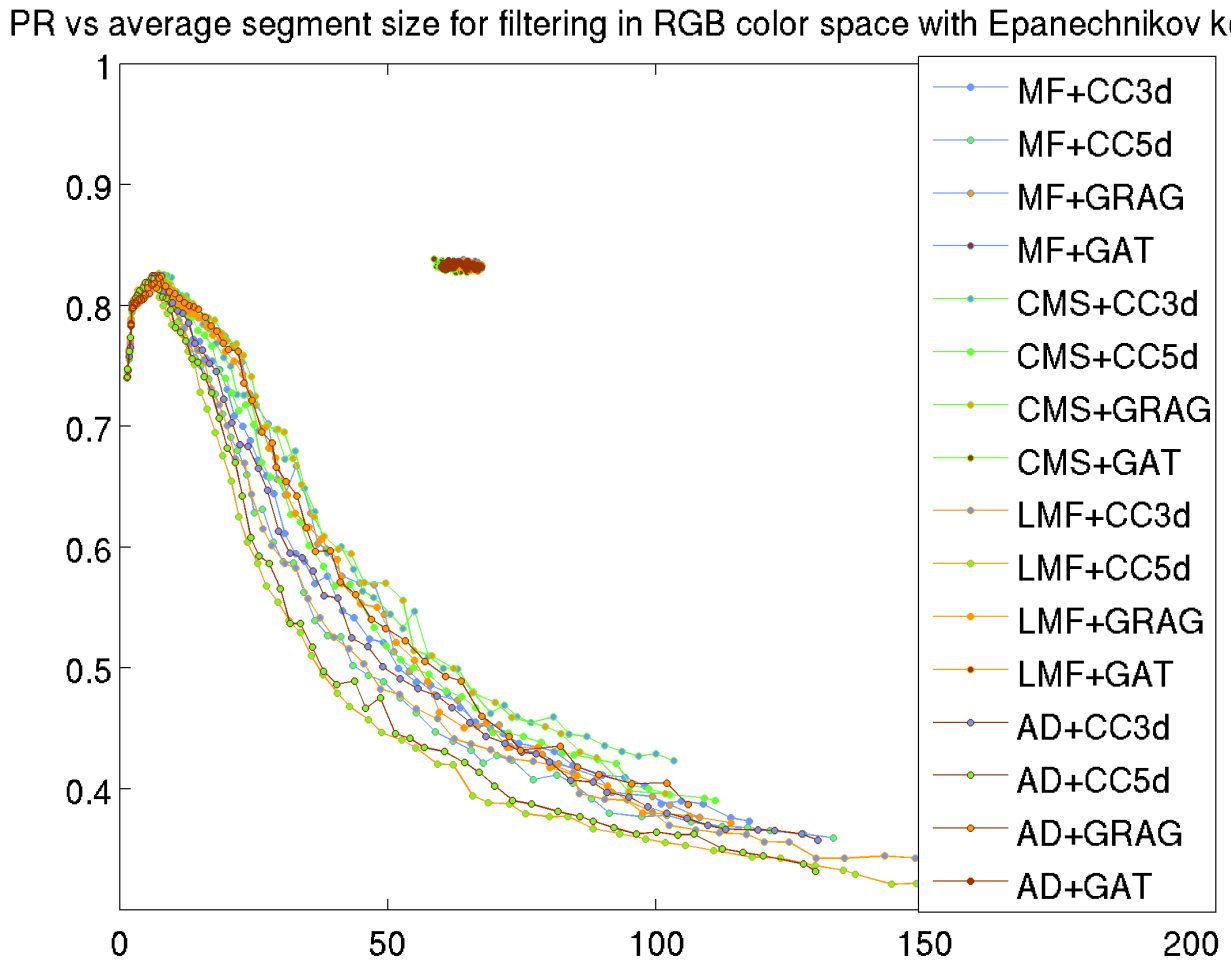


Figure 3.7: Probabilistic Rand Index vs average segment size plots when filtering is performed in the RGB color space with an Epanechnikov kernel.

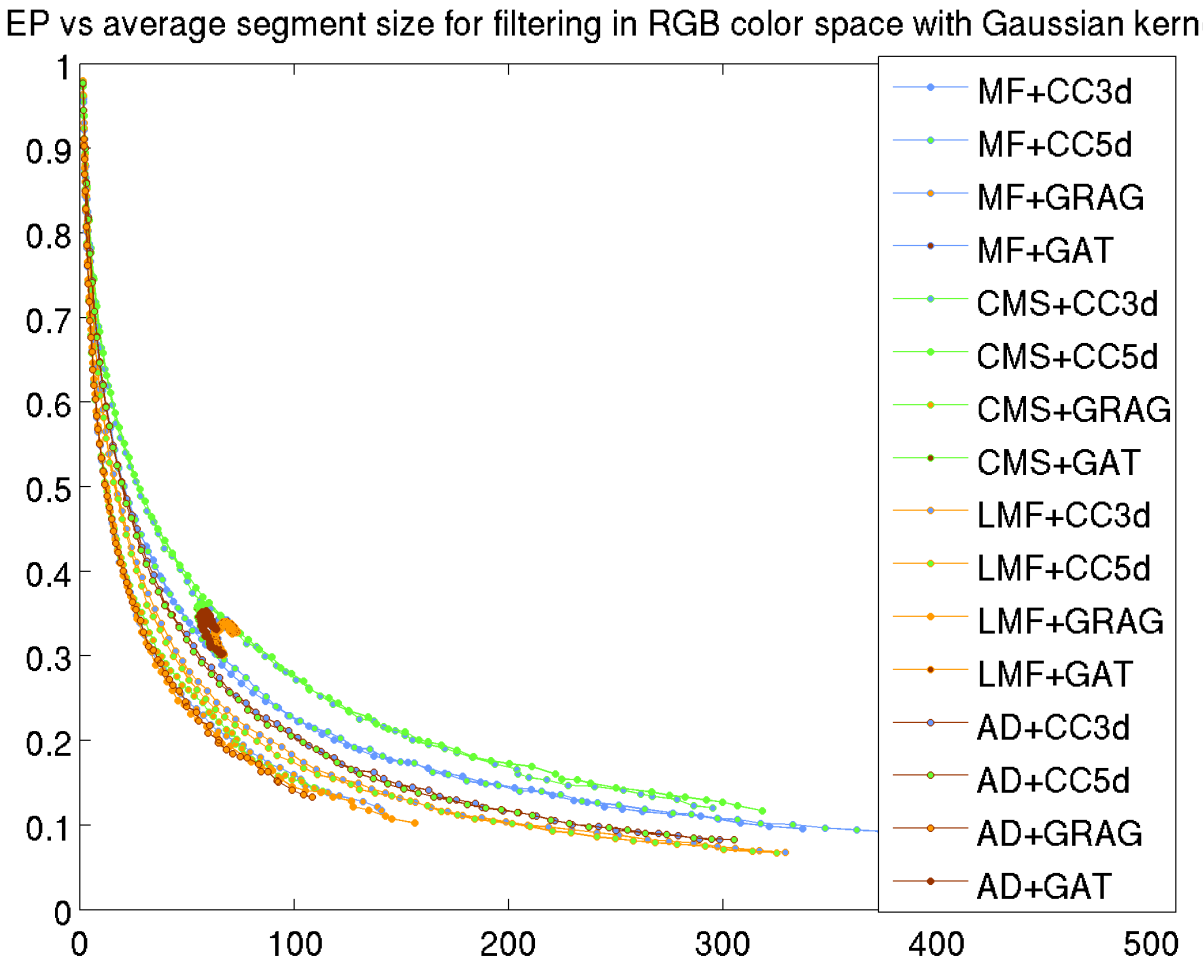


Figure 3.8: Edge Percentage vs average segment size plots when filtering is performed in the RGB color space with a Normal kernel.

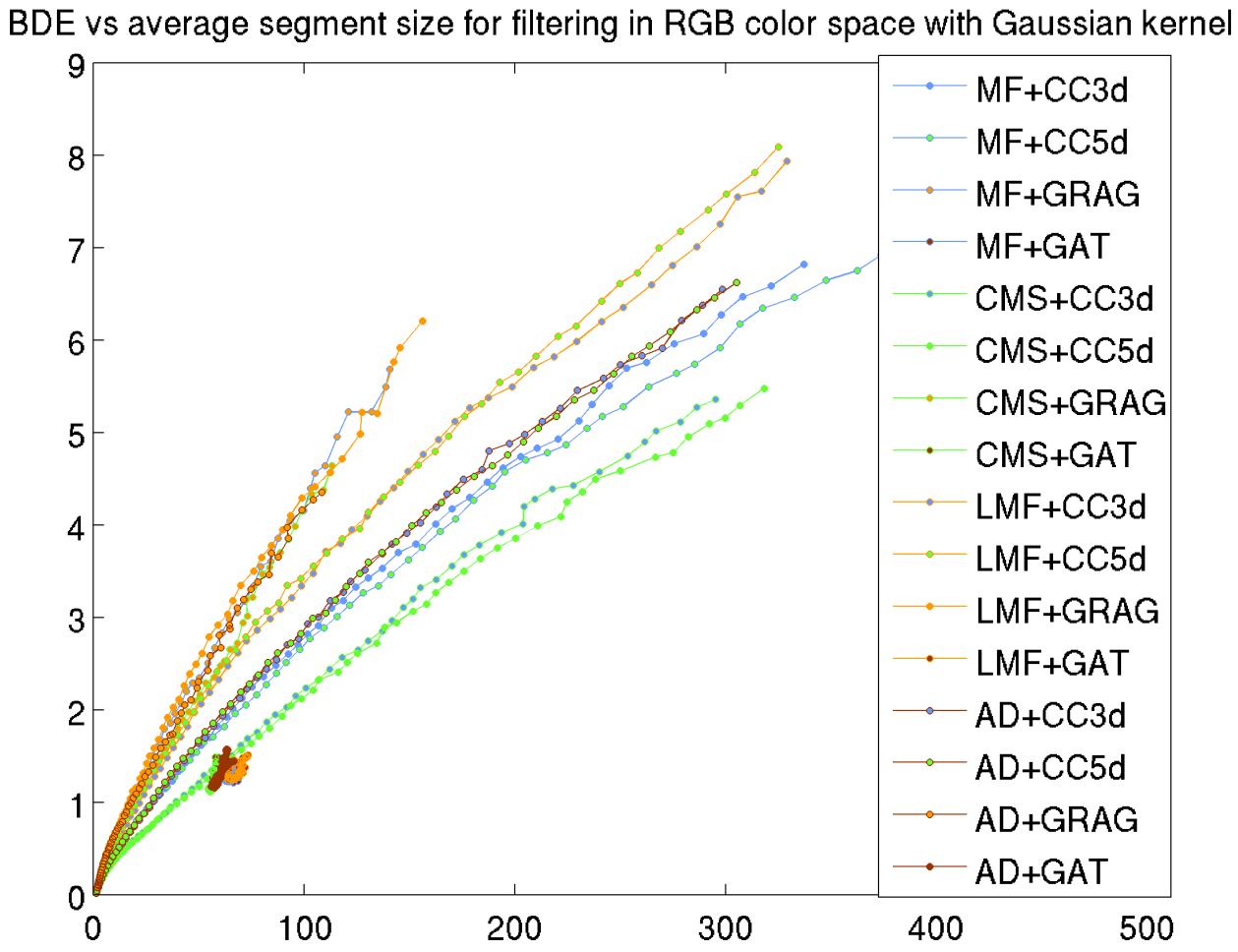


Figure 3.9: Boundary Displacement Error vs average segment size plots when filtering is performed in the RGB color space with a Normal kernel.

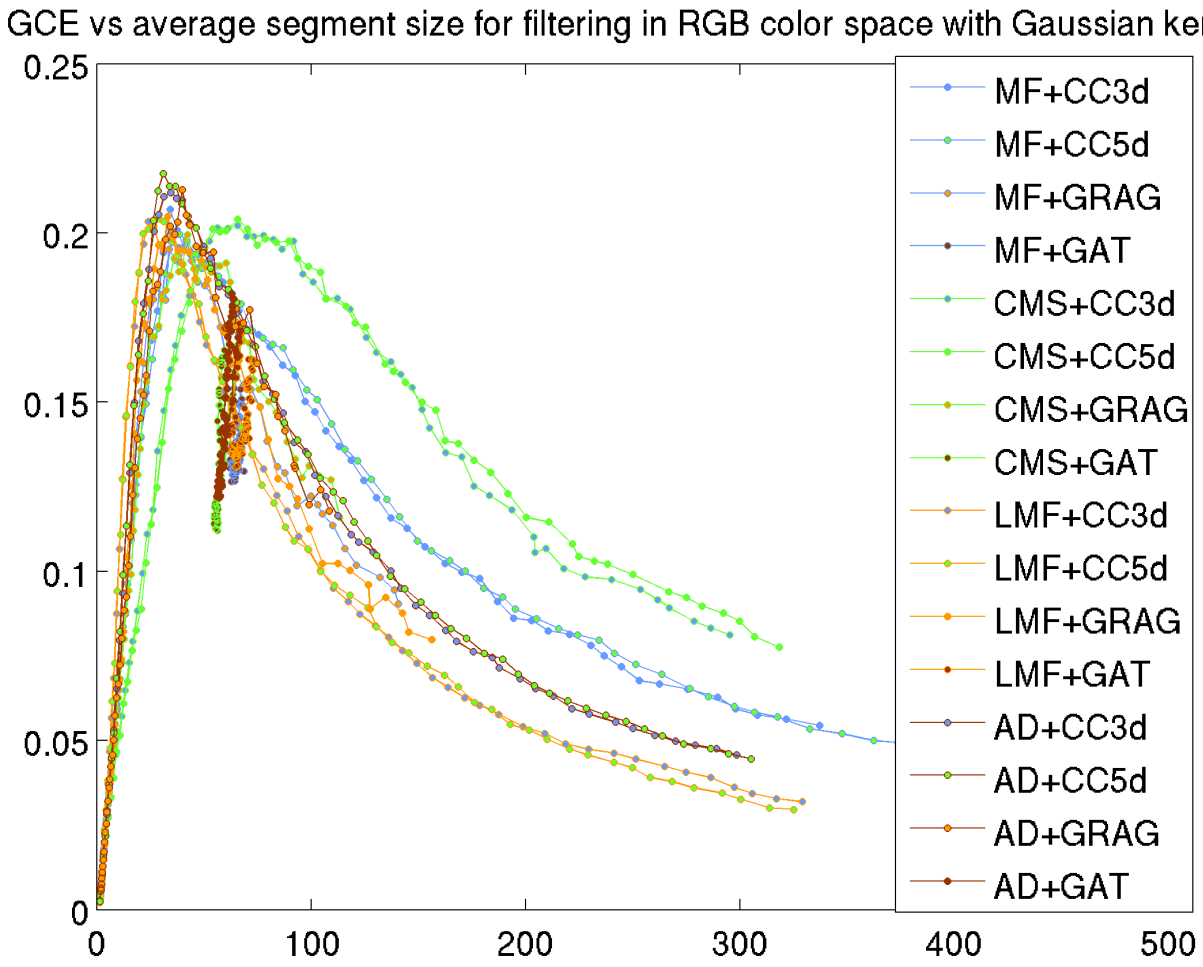


Figure 3.10: Global Consistency Error vs average segment size plots when filtering is performed in the RGB color space with a Normal kernel.

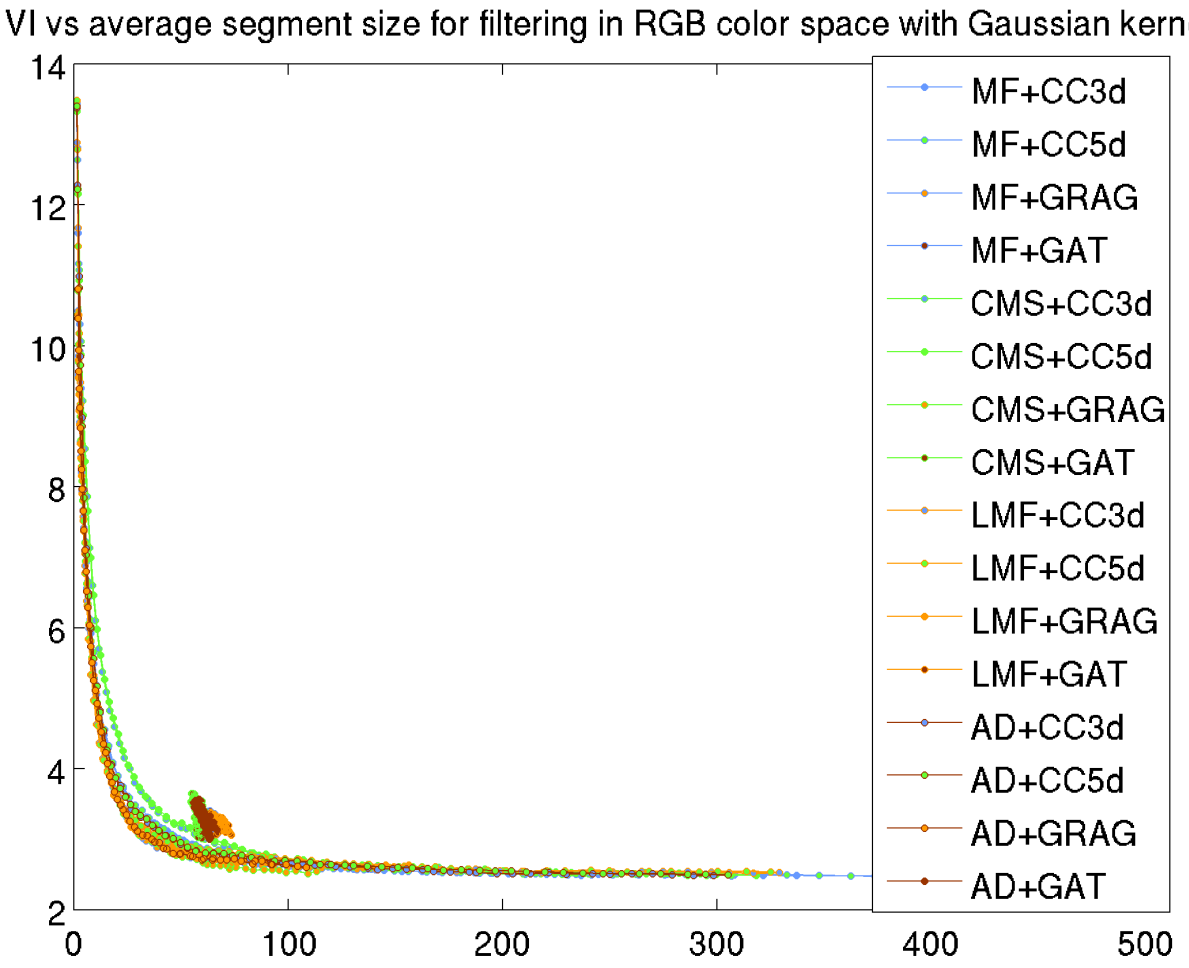


Figure 3.11: Variation of Information vs average segment size plots when filtering is performed in the RGB color space with a Normal kernel.

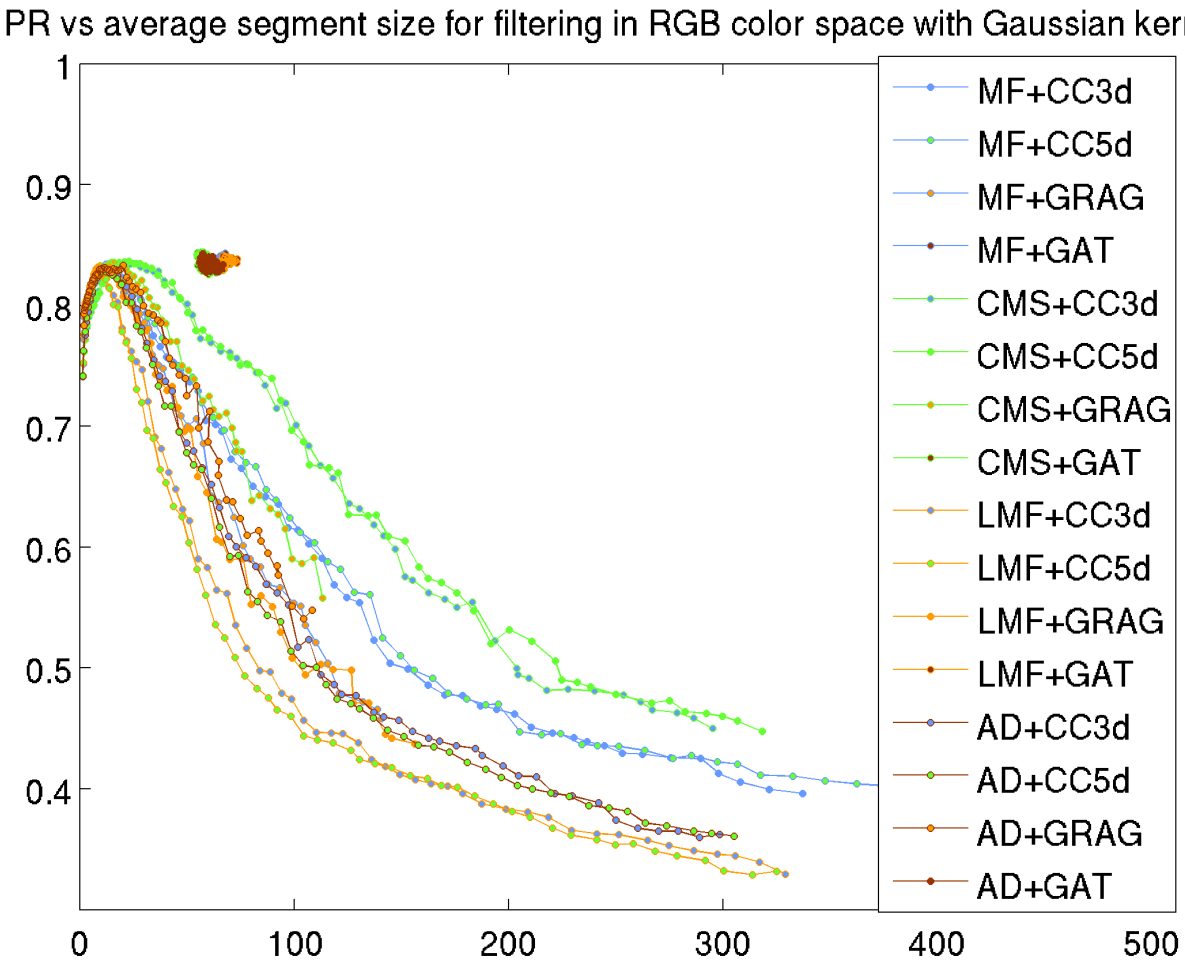


Figure 3.12: Probabilistic Rand Index vs average segment size plots when filtering is performed in the RGB color space with a Normal kernel.

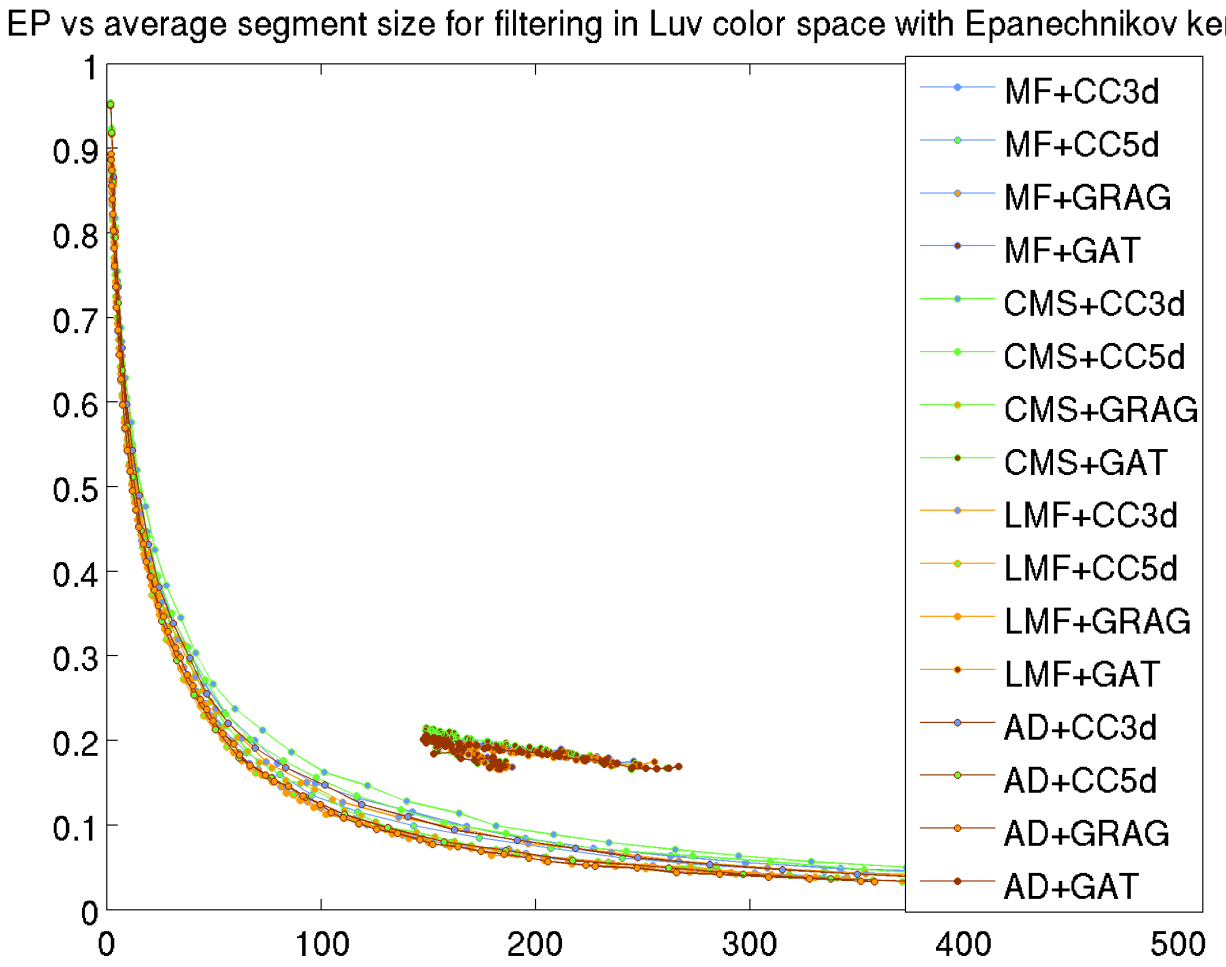


Figure 3.13: Edge Percentage vs average segment size plots when filtering is performed in the Luv color space with an Epanechnikov kernel.

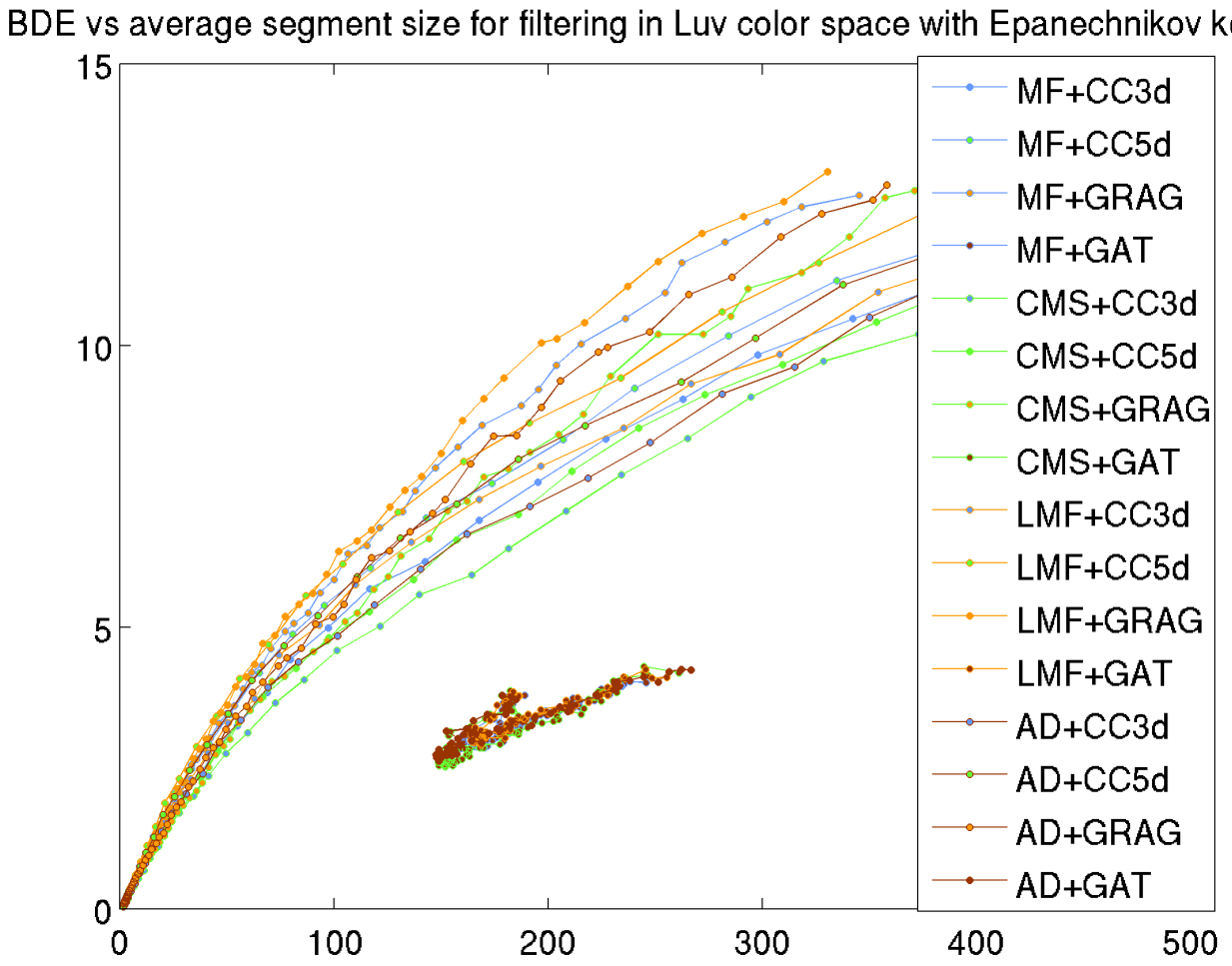


Figure 3.14: Boundary Displacement Error vs average segment size plots when filtering is performed in the Luv color space with an Epanechnikov kernel.

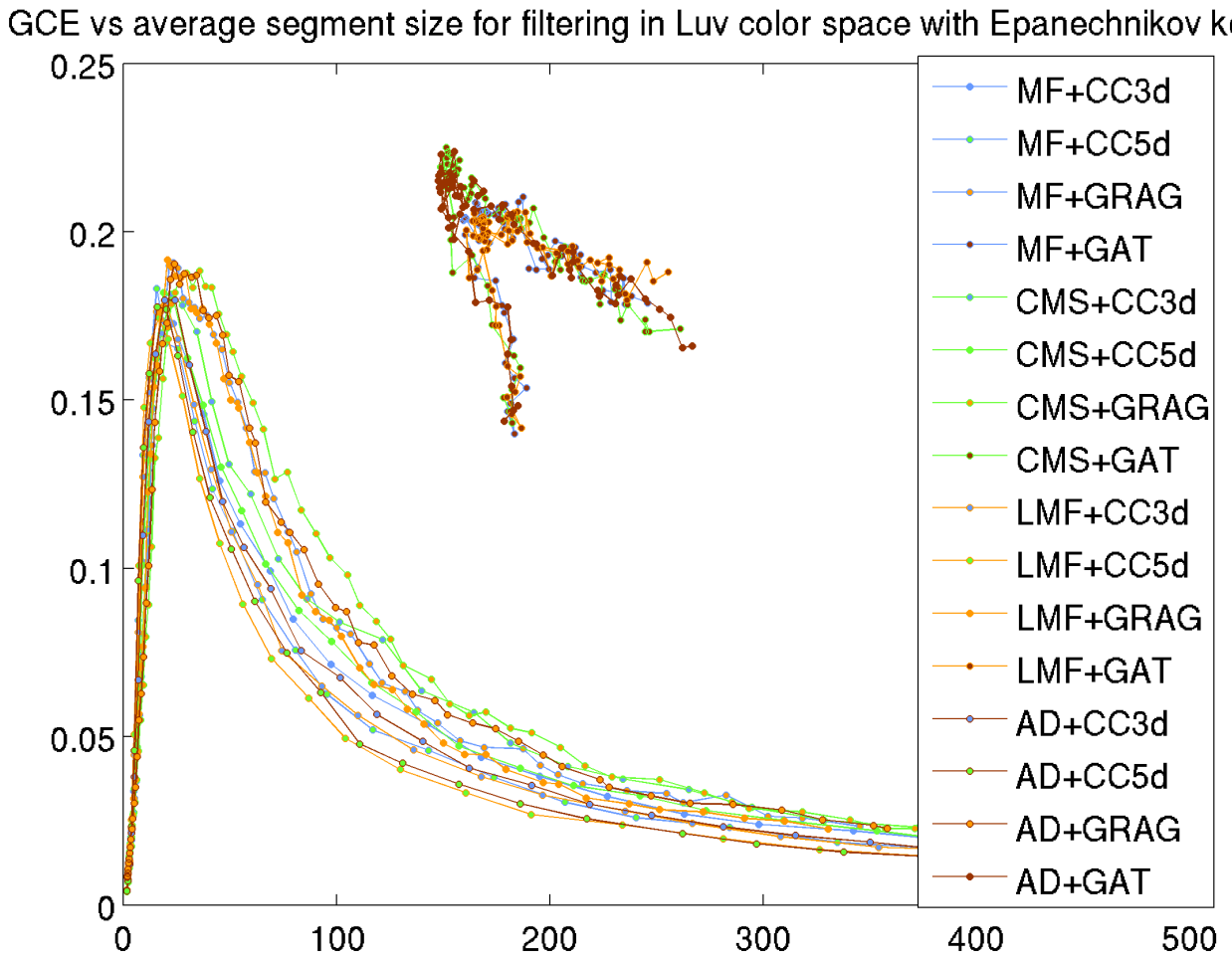


Figure 3.15: Global Consistency Error vs average segment size plots when filtering is performed in the Luv color space with an Epanechnikov kernel.

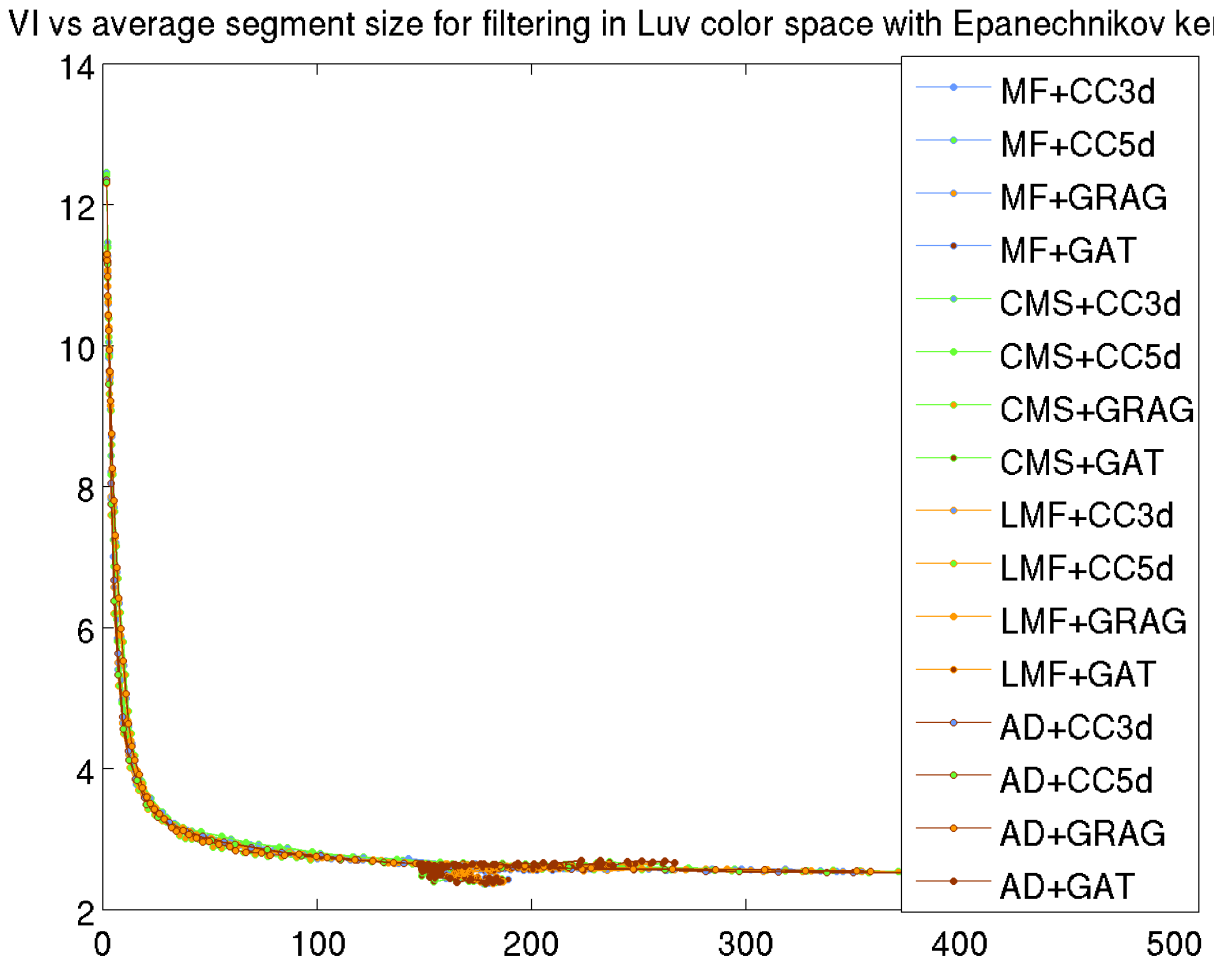


Figure 3.16: Variation of Information vs average segment size plots when filtering is performed in the Luv color space with an Epanechnikov kernel.

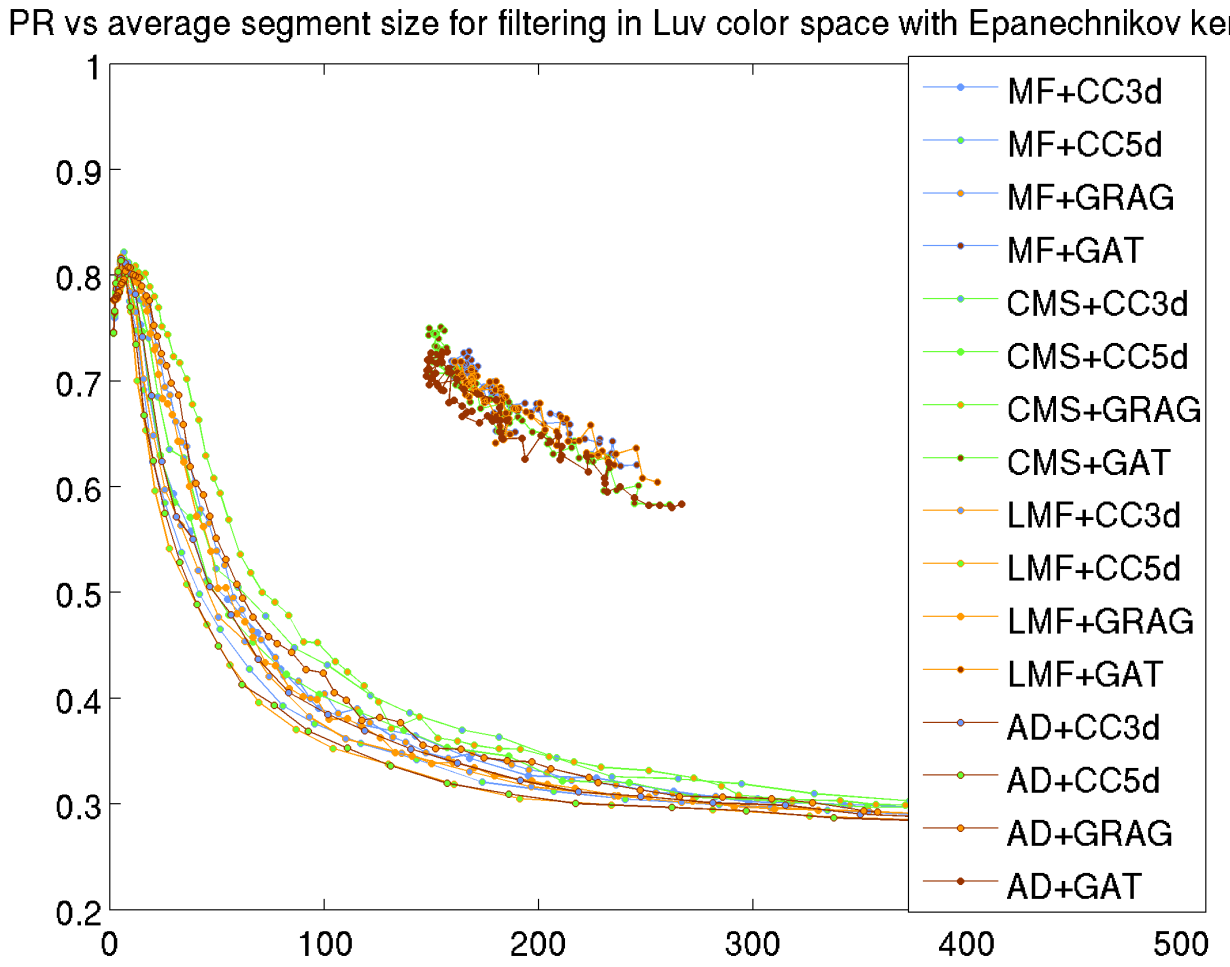


Figure 3.17: Probabilistic Rand Index vs average segment size plots when filtering is performed in the Luv color space with an Epanechnikov kernel.

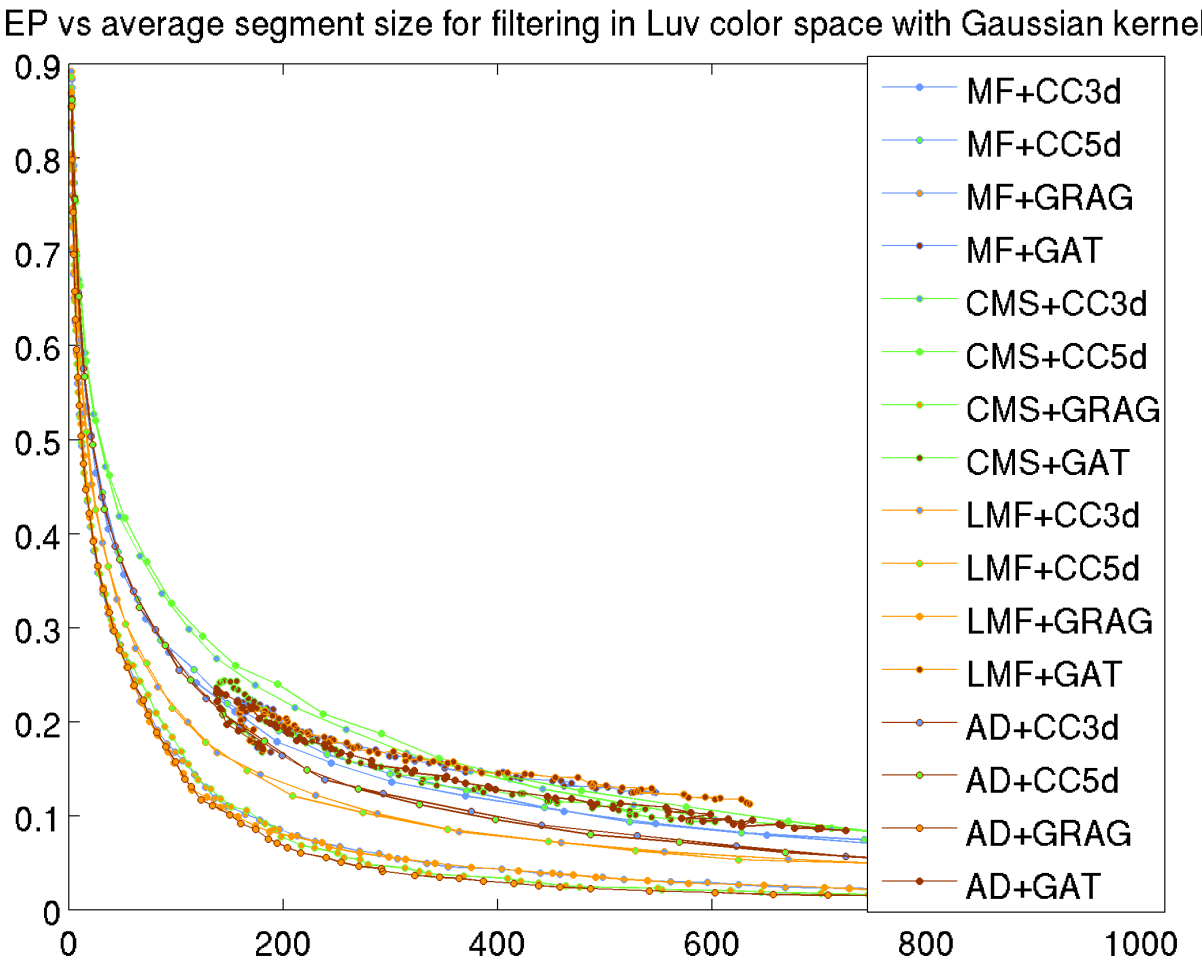


Figure 3.18: Edge Percentage vs average segment size plots when filtering is performed in the Luv color space with a Normal kernel.

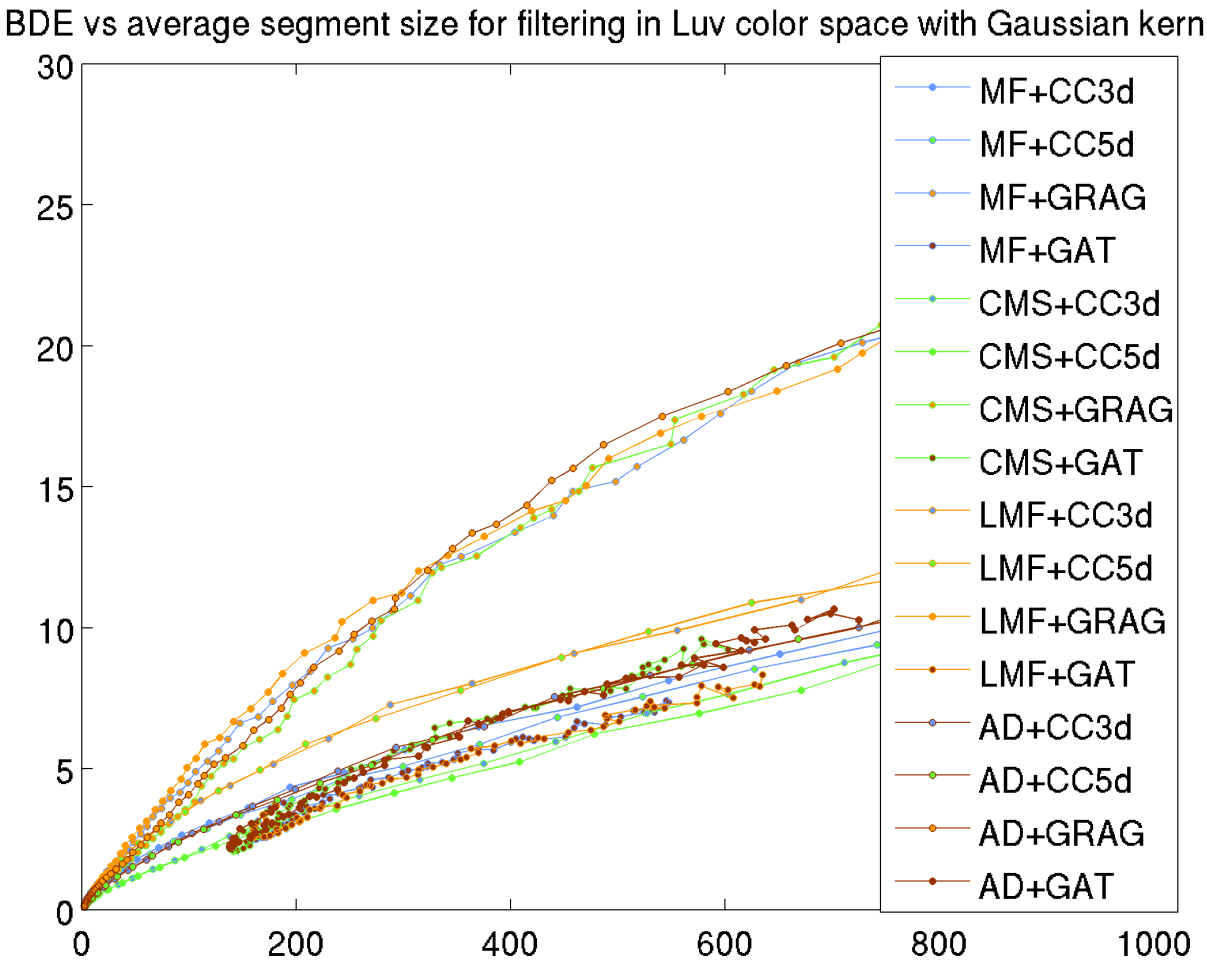


Figure 3.19: Boundary Displacement Error vs average segment size plots when filtering is performed in the Luv color space with a Normal kernel.

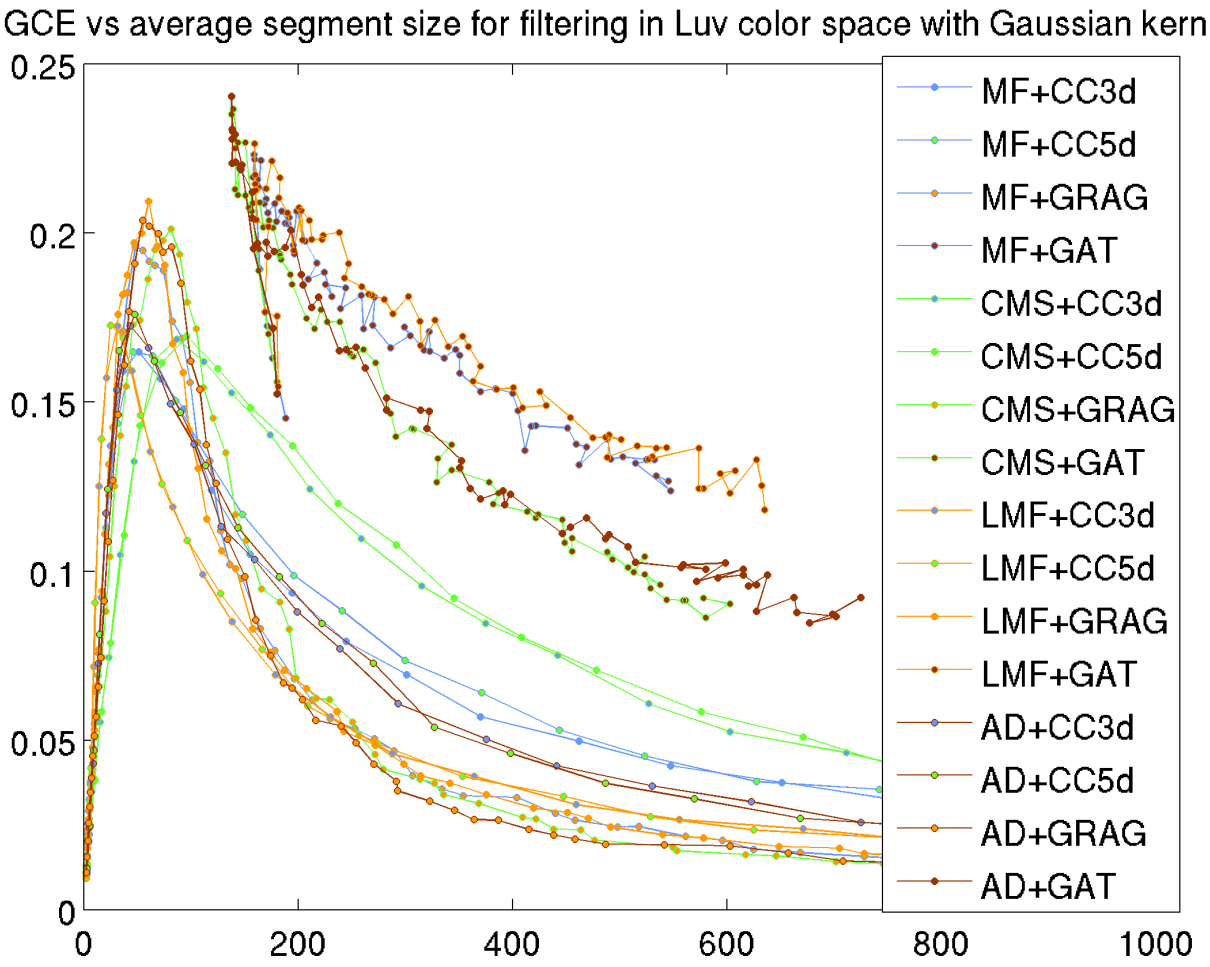


Figure 3.20: Global Consistency Error vs average segment size plots when filtering is performed in the Luv color space with a Normal kernel.

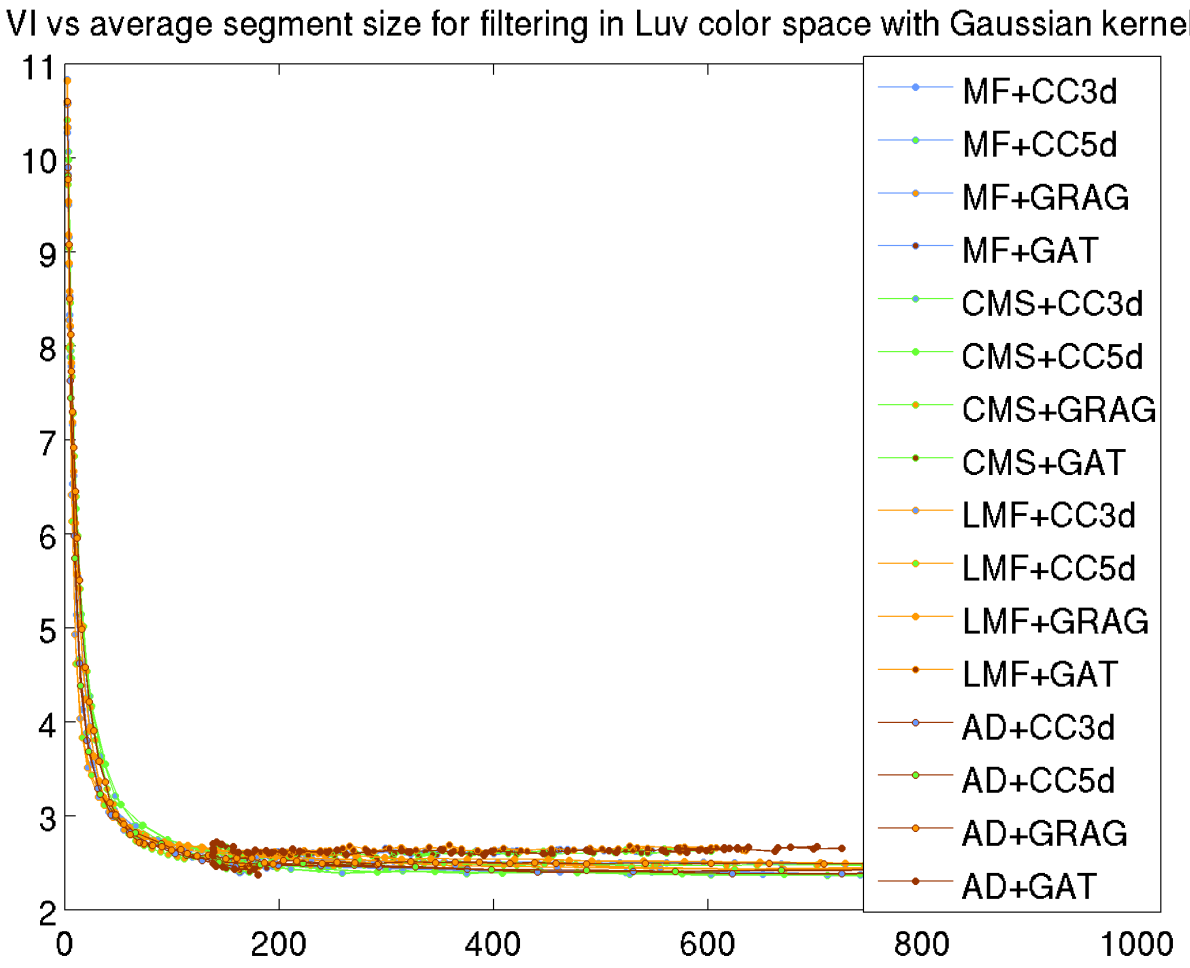


Figure 3.21: Variation of Information vs average segment size plots when filtering is performed in the Luv color space with a Normal kernel.

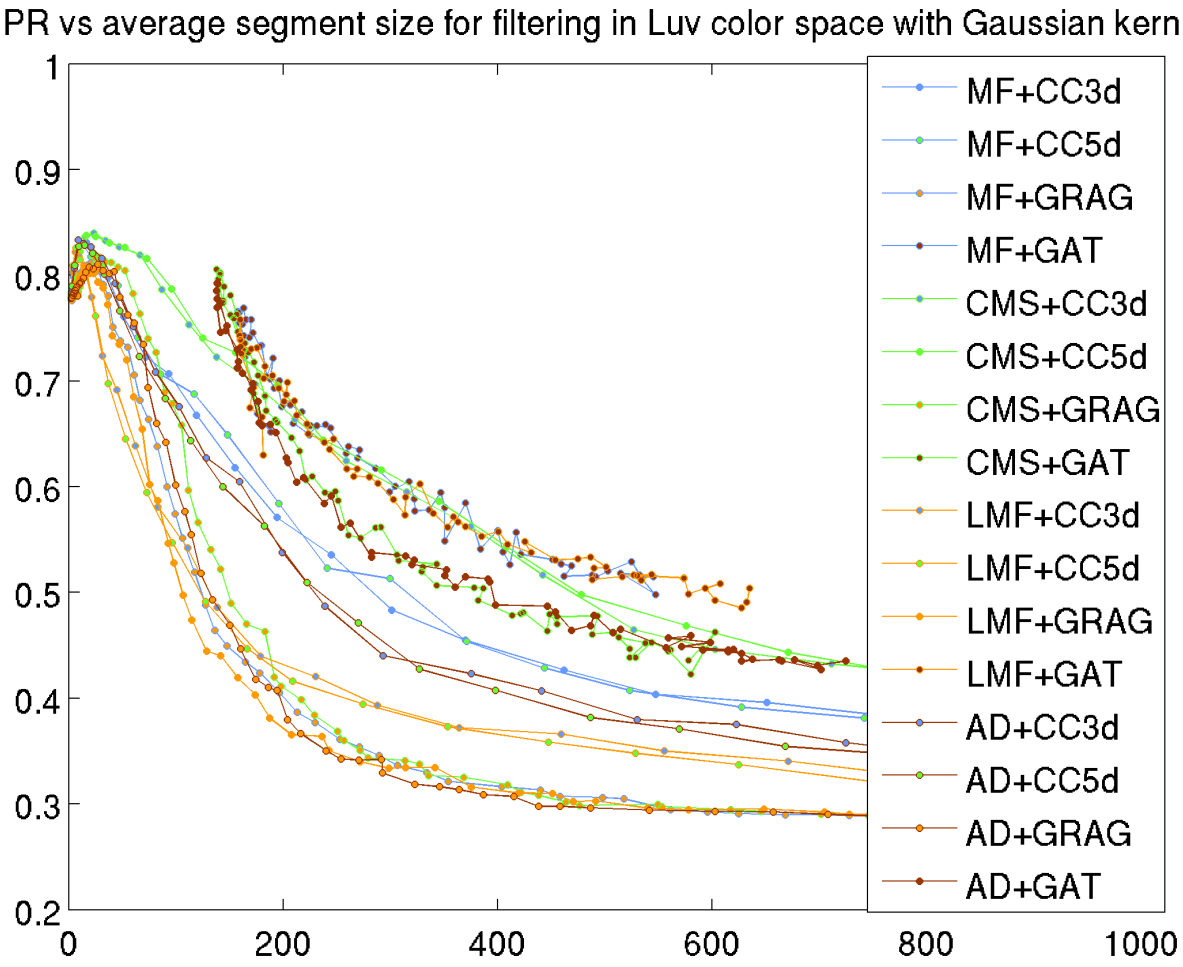


Figure 3.22: Probabilistic Rand Index vs average segment size plots when filtering is performed in the Luv color space with a Normal kernel.

In the figures above all the plots display the average values over the whole database of segmentations. Before proceeding with the analysis of the results we want to emphasize that there is a high variability in the results for individual images. The grouping method of the adaptive threshold (GAT) exhibits the lowest inter-image variability no matter which filtering method it is coupled with. All the other grouping methods are highly sensitive on the image to be segmented. The same observation was mentioned by Unnikrishnan et al. in [40] where they compared Comaniciu and Meer mean shift method against the segmentation based on GAT.

All the segmentation methods based on GAT grouping are non monotonic. While all other segmentation methods produce curves that are either piecewise monotonically decreasing or increasing (depending on the measure), the curves of GAT methods manifest an unpredictable non monotonic behavior. This is best displayed in Figs. 3.17, 3.22. The curve for the LMF+GAT method, for example, in Fig. 3.22 not only is non decreasing, indicating that successive values of the color bandwidth might produce either better or worse results, but also it might lead to smaller or larger average segment sizes. The cause of this behavior is the adaptive threshold used for grouping. It is well documented that the merging of two regions in GAT grouping is decided based on the inter-region and intra-region edge distribution. Since anisotropic filtering smooths some edges while keeps other intact, if the inter region edges between two regions are smoothed more that the intra-region edges, then GAT will merge the two regions. In the opposite case, GAT will not merge the two regions. Thus, the overall segmentation is not guaranteed to be "consistent" for successive filtering values.

On average the segmentation methods based on GAT grouping outperform all the other segmentation methods. More specifically, they display the most similarity with the human segmentations while the average segment size is larger than the other segmentation methods. For example, all the GAT based methods in Fig. 3.17 form a cluster with significantly larger Probabilistic Rand Index values that the rest of the methods. Still, it is not clear which combination of filtering method should be used with the GAT algorithm to obtain the best results. We will investigate this topic further in the next section.

On average the segmentation methods based on GAT grouping exhibit the least variation of the average segment size i.e., in a sense they are the most stable to color resolution changes. For filtering in Luv color space with an Epanechnikov kernel, for example, all the other methods produce average segments varying from 2 to ~ 400 pixels, while GAT methods give results from ~ 140 to ~ 270 pixels. This is related to the use of the constant value $k = 500$ for the GAT algorithm.

The segmentation methods based on CC3D and CC5D grouping exhibit very similar performance, with the CC3D ones producing slightly better segmentation results. This indicates that there is an advantage performing the grouping in the color dimensions only, opposed to the case of filtering where 5D filtering gives better results. The GRAG based methods in some settings (i.e., color space and kernel function combinations) outperform the CC3D and CC5D methods, while in other settings perform equally well or even worse.

The Global Consistency Error (GCE) graphs prove once more what is theoretically predicted i.e., this measure only makes sense when the number of segments in

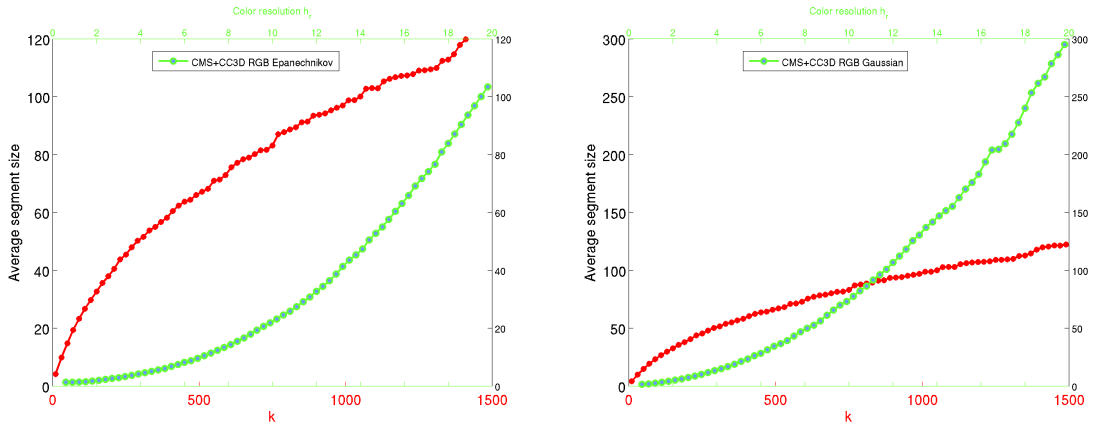
the human and computer segmentation is comparable. In our setting this requirement is only satisfied for a small range of color segmentations, hence these graphs are misleading. That is why we obtain a value close to 0 for very small and very large color resolutions.

The graphs of the Variation of Information (VI) measure are the least discriminative, because all the plots converge very rapidly to a value of ~ 3. In the subsequent comparisons we will not use the Variation of Information (VI), Edge Percentage (EP) and Global Consistency Error (GCE) graphs.

The graphs of the Probabilistic Rand Index (PR) and Boundary Displacement Error (BDE) measures are the most discriminative. So in the next sections we will use these for comparing the different segmentation methods.

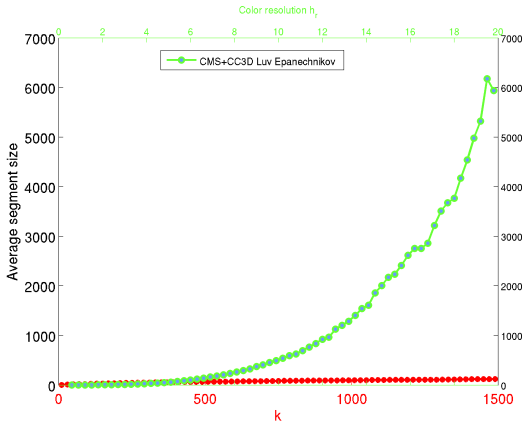
Color Mean Shift (CMS) based segmentation methods outperform all the other filtering methods when coupled with the same grouping methods. This indicates that the better filtering results produces by CMS lead to better segmentation results.

We previously mentioned (Sec. 2.4) that filtering in Luv color space produces smoother images, for a given color resolution, compared to filtering in RGB. As a consequence the average segment size is an order of magnitude larger as a careful examination of the X axis of the plots reveals. The kernel selection also affects the average segment size. Use of a Normal kernel leads to larger segments compared to Epanechnikov kernel, as expected.

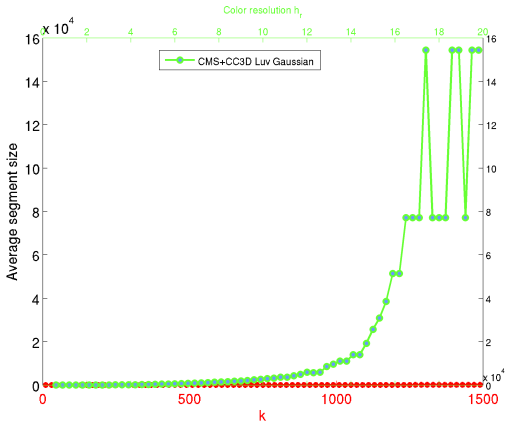


(a) RGB, Epanechnikov

(b) RGB, Normal



(c) Luv, Epanechnikov



(d) Luv, Normal

Figure 3.23: In red and green, we plot the average segment size as a function of the segmentation parameter k in the case of GAT or color resolution h_r in the case of CC3D, respectively.

3.4.3 Adjusting the threshold parameter (k) of the GAT grouping method

So far we used the Grouping with an Adaptive Threshold method of Felzenszwalb and Huttenlocher [7] with a fixed value for the threshold parameter $k = 500$. As we showed in the previous section this leads to small variability in the average segment size not matter how much we smooth the image in advance. In this section we explore the idea of changing the grouping parameter k according to the filtering value.

First, in Fig. 3.23 we plot the average segment size with respect to the value of k. In the same graphs we display the average segment size obtained with a combination of CMS and CC3D methods for different values of color resolution and for different color space and kernel function settings. All the results are computed for the whole database of images. As we observe in the plots, the average segment size increase is much smoother for the GAT method compared to all the other segmentation methods, especially these that perform the filtering in the Luv color space.

In Figs. 3.24, 3.25 we display the implicit BDE and Rand Index values for the GAT method with respect to the average segment size, respectively. We compare the results with the CMS+CC3D segmentation method for different color space and kernel combinations. One can easily verify that the GAT method performs slightly worse, under the BDE measure, than the CMS+CC3D method, if the filtering is performed in the Luv color space with a Normal kernel. Considering the PR measure, the GAT method performs worse for small values of average segment size, but outperforms the CMS+CC3D method for larger values of average segment size.

Depending on the specific image and application, a different kind of segmentation (i.e., different number of segments) is desirable. For example, state of the art stereo algorithms [1], [2] initially perform a color-based segmentation of the image into regions with (hopefully) consistent disparities. In order to minimize the risk of grouping pixels belonging to different objects together, they perform an over-segmentation into many small segments. Shape-based object recognition, on the other hand, requires a coarser segmentation of the image; one where all the internal

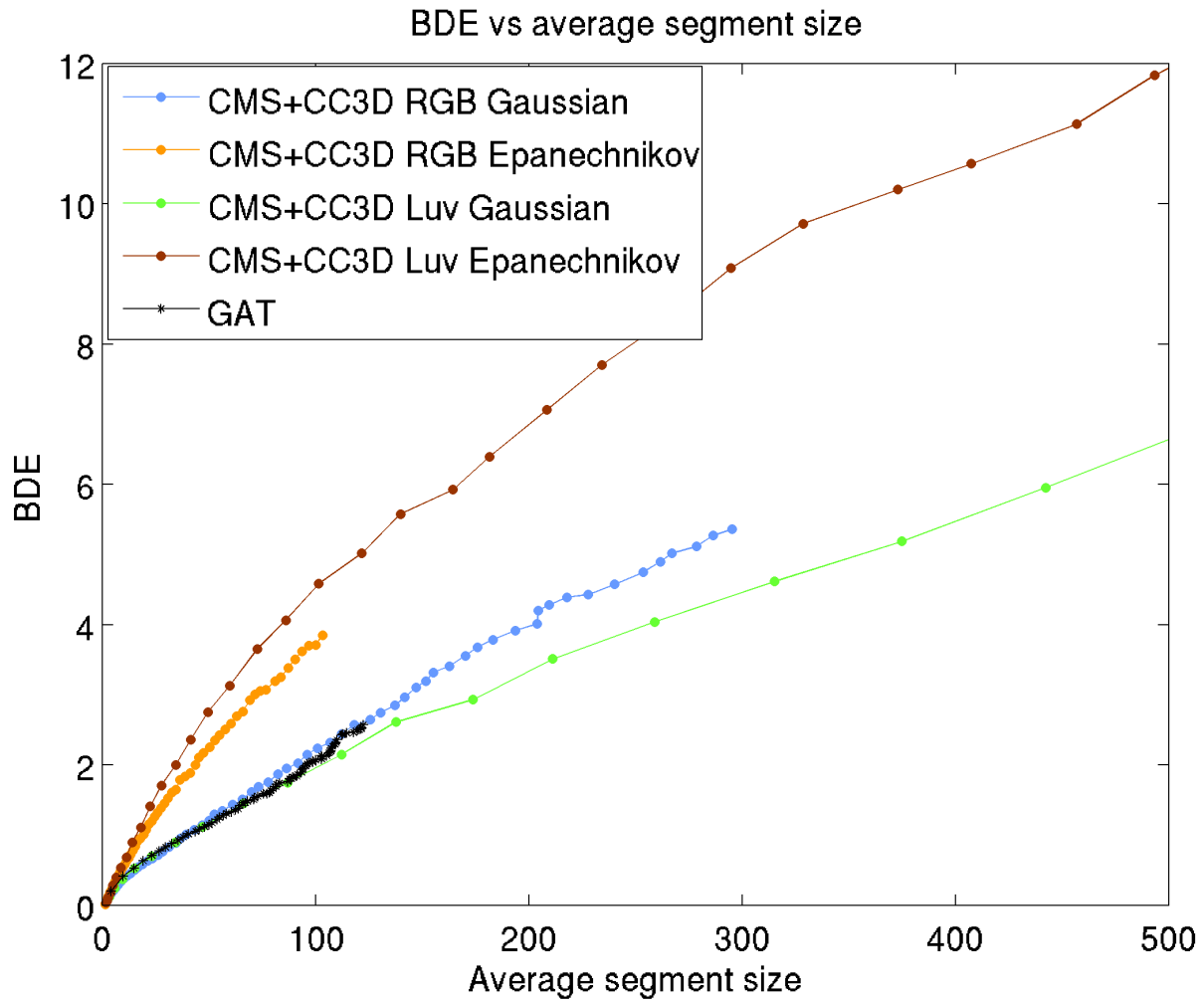


Figure 3.24: Boundary Displacement Error vs average segment size plots for the Color Mean Shift (CMS) and Connected Component in 3D (CC3D) combination and the GAT only segmentation methods.

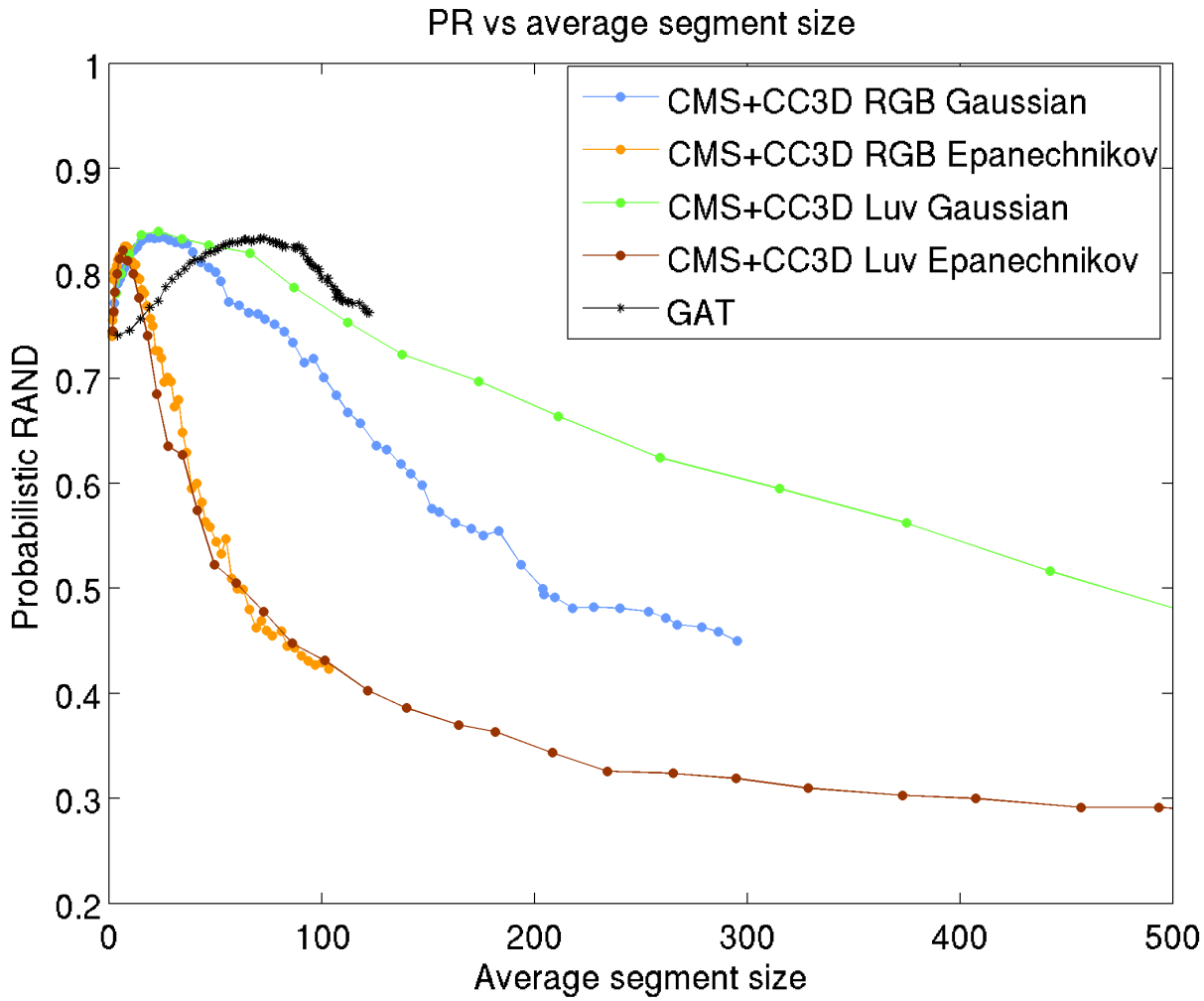


Figure 3.25: Probabilistic RAND vs average segment size plots for the Color Mean Shift (CMS) and Connected Component in 3D (CC3D) combination and the GAT only segmentation methods.

parts of an object belong to the same segment. As a consequence, it is extremely important to have a way to adjust the "granularity" of the segmentation. For all the grouping methods, but GAT, the color resolution h_r used for filtering can also be used as a segmentation threshold. As we previously discussed GAT does not use a "hard" threshold; the parameter k is used instead to control the granularity of the segmentation. From Figs. 3.23, 3.25, we observe that values of k between 170 and 1050 produce the best results, i.e. PR indices of more than 0.8 for a range of average segment size from 35 to 100 pixels. Apparently there are many ways to combine the filtering parameter h_r with the grouping parameter k. In the following experiments we use a linear relation between h_r and k[5], namely

$$k = 45.83 * h_r + 142.5. \tag{3.1}$$

In Figs. 3.26, 3.27, 3.28 and 3.29 we compare the results we obtained with the combination of all the filtering methods with the GAT grouping method. For the GAT method we display the results we obtained when we used a constant parameter $k = 500$, and when we changed the grouping parameter according to Eq. 3.1. For comparison purposes we also display the results when we used GAT grouping directly on the original images (i.e., without any filtering).

[5]We obtained the coefficients of the linear system by solving the system of (k, h_r) for values (170, 0.6) and (1050, 19.8).

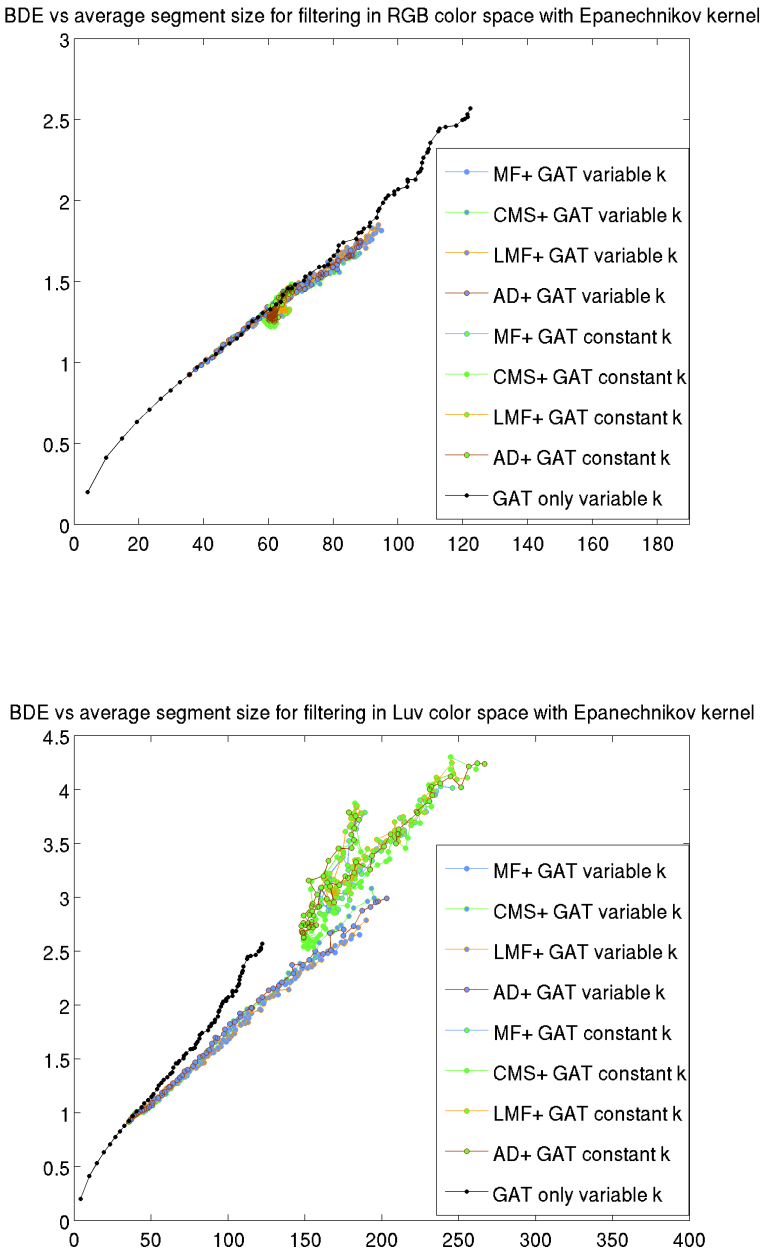


Figure 3.26: BDE vs average segment size plots for the GAT grouping method preceeded by the various filtering methods or not. We display the results for when we use a variable and a constant ($k = 500$) grouping parameter. We also display the plot when we use the GAT grouping method without filtering. In the top and bottom plots the filtering is performed on the RGB and Luv color space with an Epanechnikov kernel respectively.

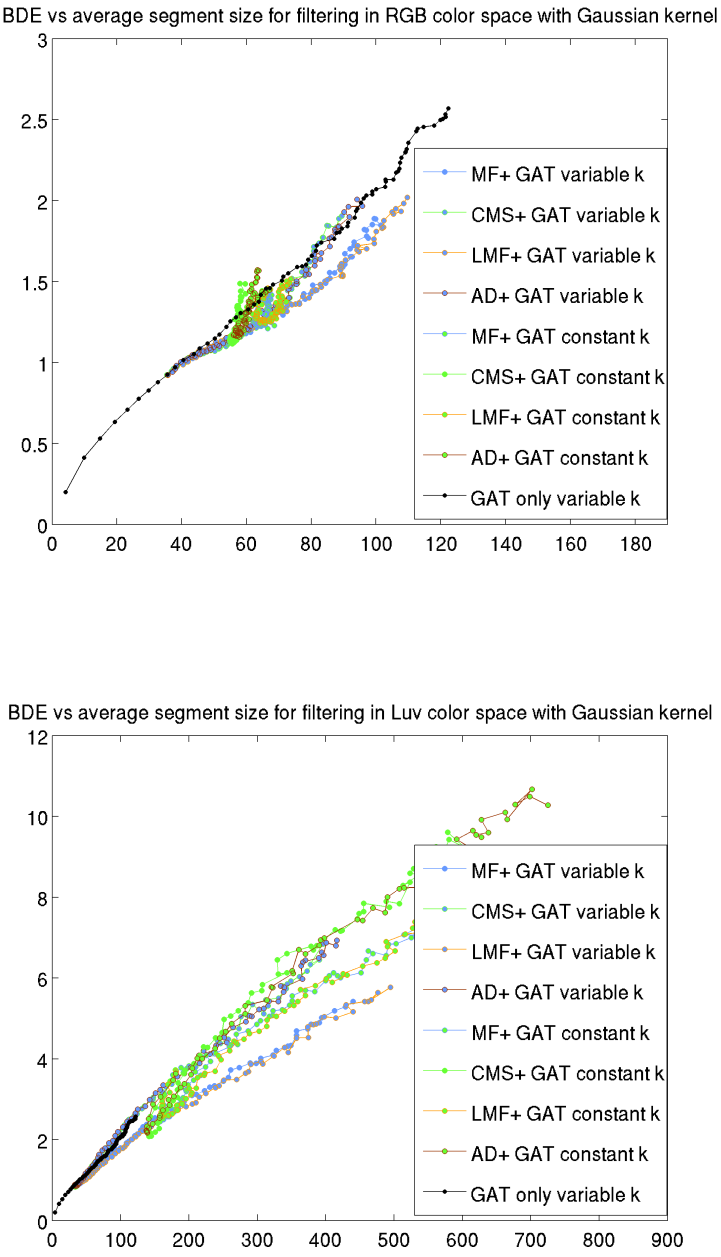


Figure 3.27: BDE vs average segment size plots for the GAT grouping method preceeded by the various filtering methods or not. We display the results for when we use a variable and a constant ($k = 500$) grouping parameter. We also display the plot when we use the GAT grouping method without filtering. In the top and bottom plots the filtering is performed on the RGB and Luv color space with a Normal kernel respectively.

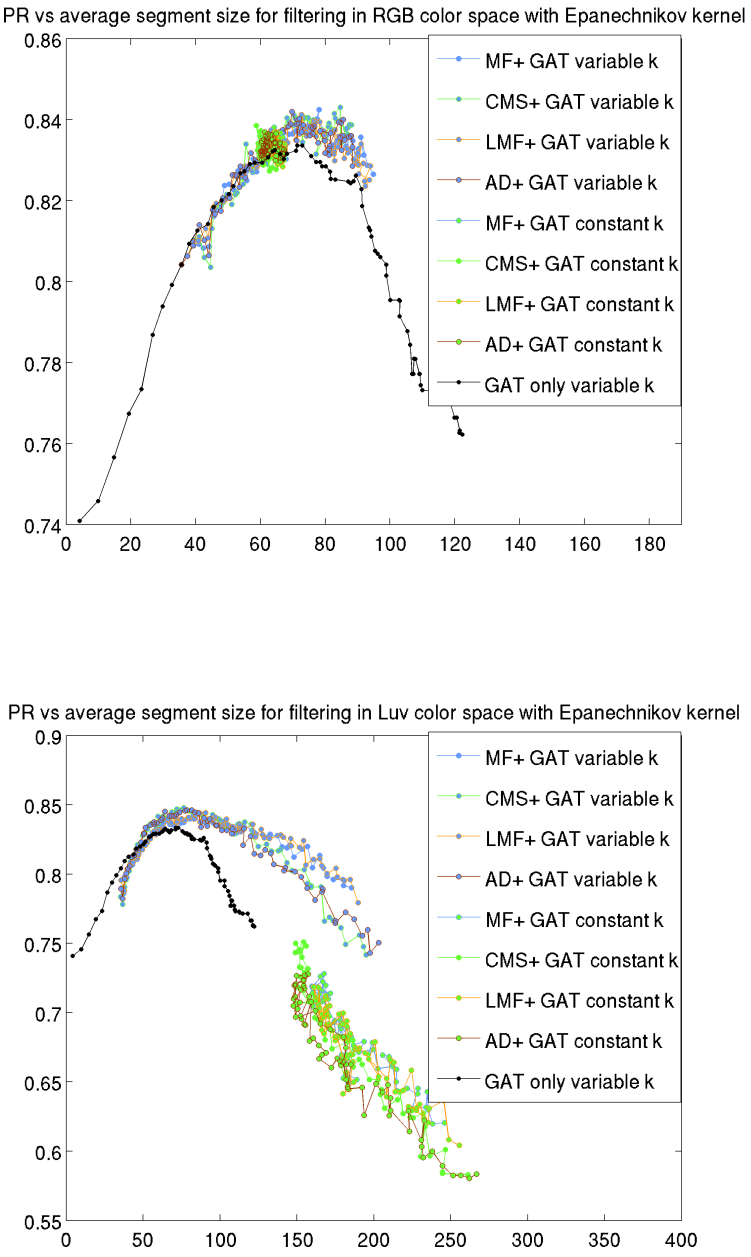


Figure 3.28: PR vs average segment size plots for the GAT grouping method preceeded by the various filtering methods or not. We display the results for when we use a variable and a constant ($k = 500$) grouping parameter. We also display the plot when we use the GAT grouping method without filtering. In the top and bottom plots the filtering is performed on the RGB and Luv color space with an Epanechnikov kernel respectively.

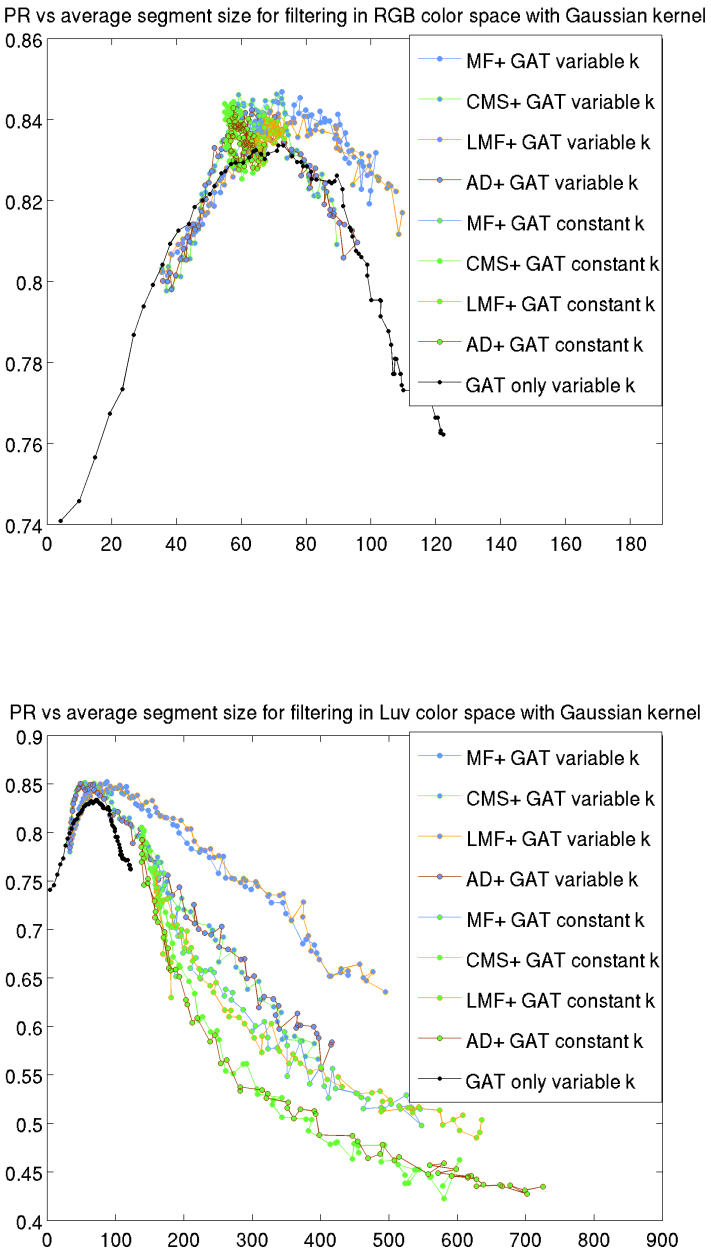


Figure 3.29: PR vs average segment size plots for the GAT grouping method preceeded by the various filtering methods or not. We display the results for when we use a variable and a constant ($k = 500$) grouping parameter. We also display the plot when we use the GAT grouping method without filtering. In the top and bottom plots the filtering is performed on the RGB and Luv color space with a Normal kernel respectively.

The first thing to notice is that the plots of the filtering+GAT grouping with a variable grouping parameter k are more "spread out" on the X-axis, meaning that they present more variability on the average segment size. This is expected since k directly affects the granularity of the segmentation. What is also expected is that filtering (in the Luv color space)+GAT grouping plots leads to larger segment sizes, compared to GAT segmentations without filtering. When the filtering was performed on the RGB color space there was little difference on the image size.

The second and most important observation from these figures is that *Mode Finding coupled with GAT grouping with a variable k outperforms all other combinations*. The second best combination is Local Mode Filtering with GAT grouping with a variable k, while both the Color Mean Shift and the Anisotropic Diffusion methods perform slightly worse. In the case of GAT grouping with a constant $k = 500$ all the filtering methods performed equally bad. Finally the GAT grouping with varying k without any filtering consistently performs worse than when we use LMF or MF filtering.

At a first glance, the outcome of these experiments might seem contradicting; the less filtering one performs the better the results are, while no filtering at all still gives bad results. There is a very intuitive explanation of this phenomenon, though, if the details of the grouping algorithm are considered. GAT adjusts the threshold for merging regions based on the inter-region and intra-region variability. As we showed in Sec 2.4.3 CMS and AD filtering methods produce much more uniform regions, compared to MF and LMF. As a consequence there is little intra-region variability and the merging process is disrupted.

The previous graphs makes one wonder how the segmentation results would be if we use Bilateral Filtering (BF) instead of MF or LMF. In essence, Bilateral Filtering is equivalent to LMF with the maximum number of iterations for the optimization problem limited to 1. The next figures show the results of BF coupled with GAT (with varying k).

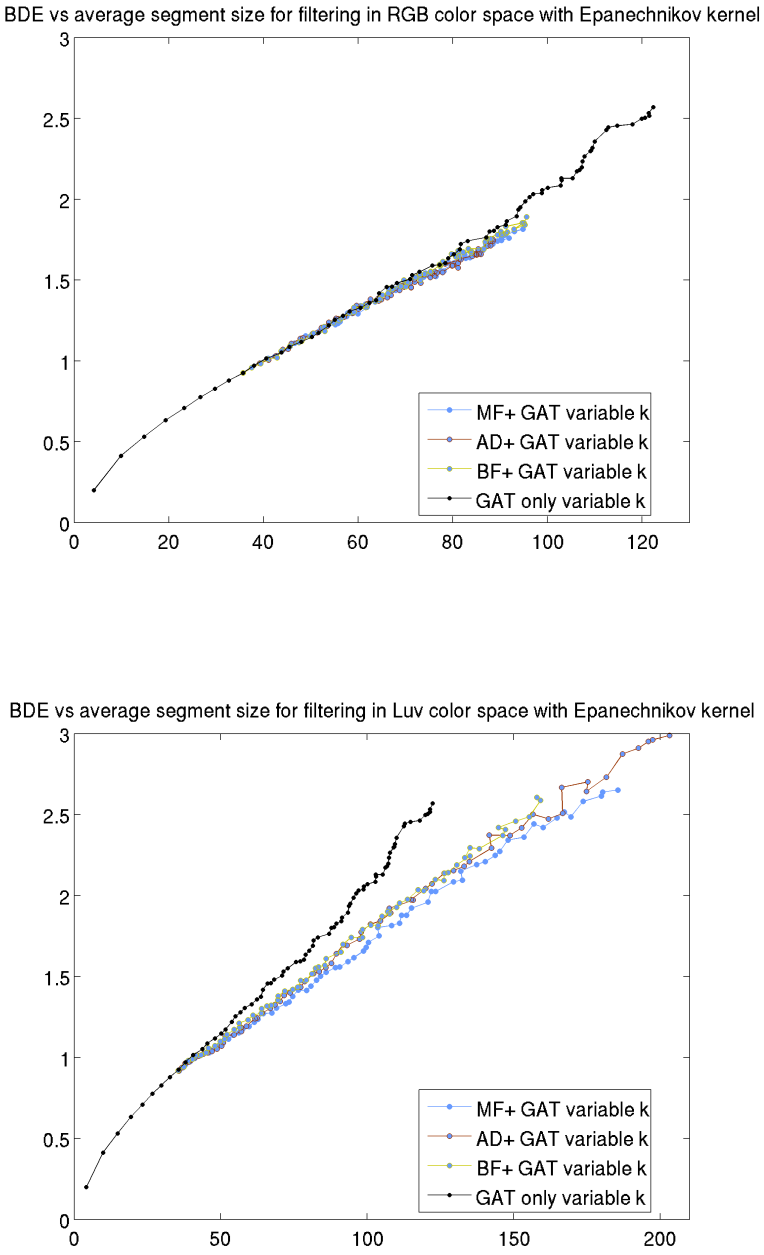


Figure 3.30: BDE vs average segment size plots for Bilateral Filtering+GAT with varying k. For comparison we also present the results of MF+GAT, AD+GAT and GAT only. In the top and bottom plot the filtering is performed on the RGB and Luv color space respectively. In all the methods an Epanechnikov kernel is used.

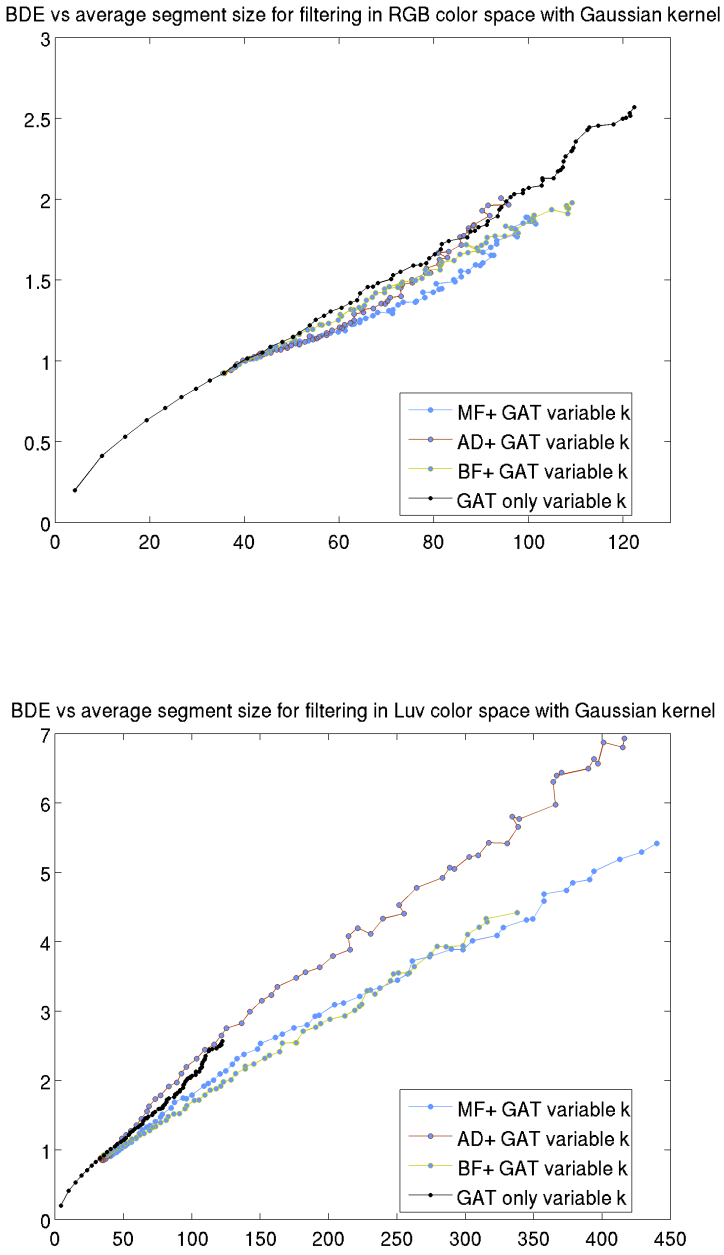


Figure 3.31: BDE vs average segment size plots for Bilateral Filtering+GAT with varying k. For comparison we also present the results of MF+GAT, AD+GAT and GAT only. In the top and bottom plot the filtering is performed on the RGB and Luv color space respectively. In all the methods a Normal kernel is used.

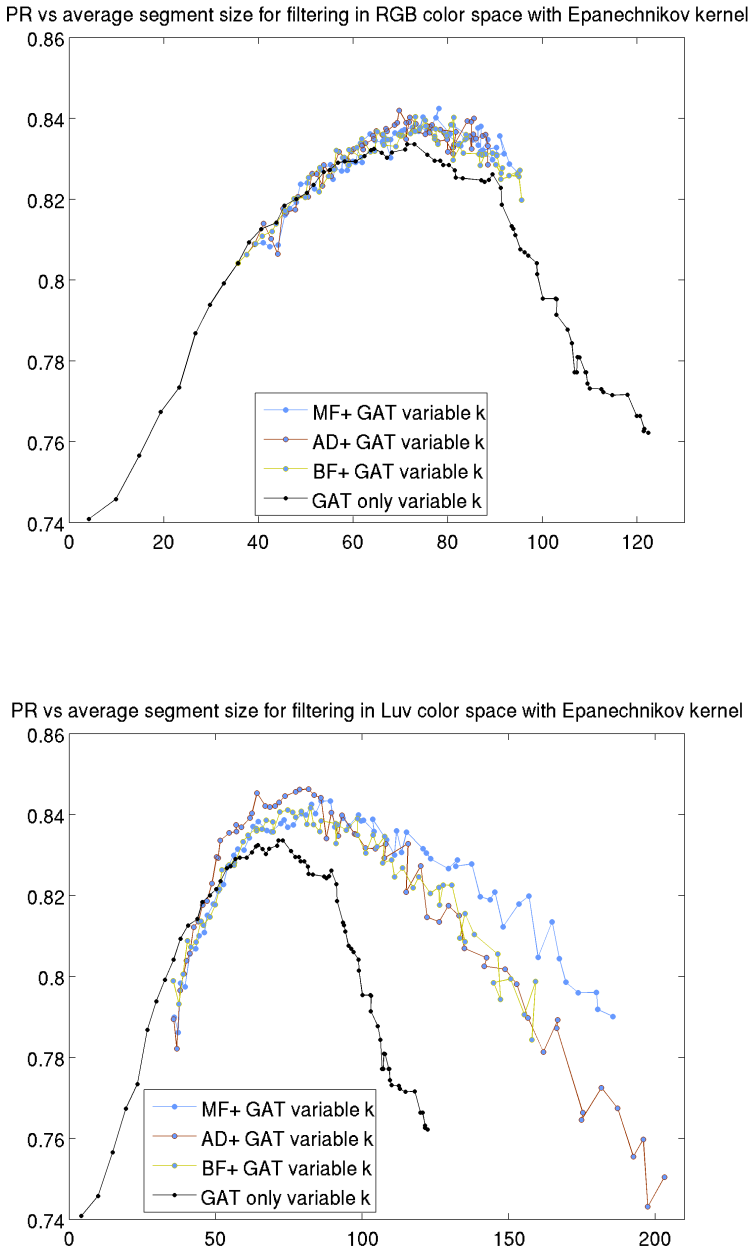


Figure 3.32: PR vs average segment size plots for Bilateral Filtering+GAT with varying k. For comparison we also present the results of MF+GAT, AD+GAT and GAT only. In the top and bottom plot the filtering is performed on the RGB and Luv color space respectively. In all the methods an Epanechnikov kernel is used.

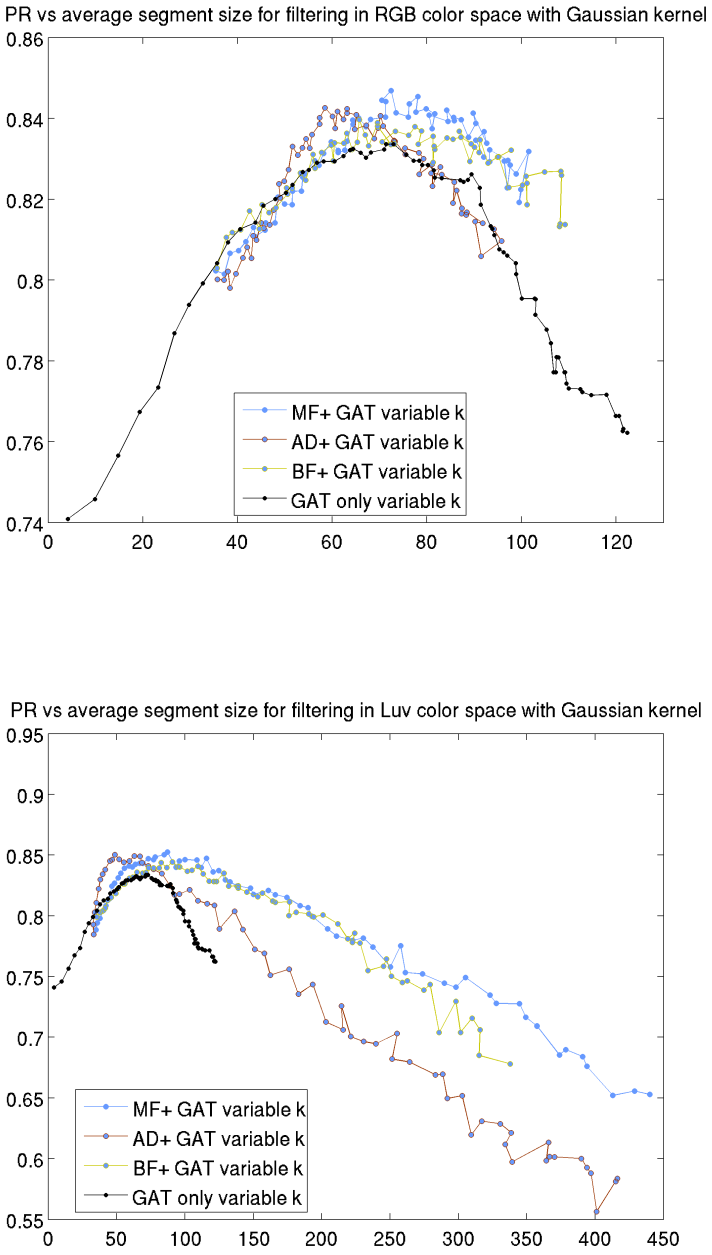


Figure 3.33: PR vs average segment size plots for Bilateral Filtering+GAT with varying k. For comparison we also present the results of MF+GAT, AD+GAT and GAT only. In the top and bottom plot the filtering is performed on the RGB and Luv color space respectively. In all the methods a Normal kernel is used.

In most cases BF performs slightly worse than MF and better than LMF. Especially when the filtering is performed on Luv with a Normal kernel, BF is equally good (or better) than MF. Furthermore it is multiple times faster than MF, making it the method of choice if speed is an issue. It would be interesting, as future work, to further study the interaction between the grouping parameter k and the color resolution h_r of the filtering methods.

3.4.4 Compare segmentations for filtering+grouping and grouping only methods

In the previous section we presented the results obtained with the GAT method only and compared them to the ones when the images are filtered first. In this section we present the results of grouping with and without filtering for the remaining three methods. In order to improve the quality of the figures we omit the plots for the anisotropic diffusion and local mode filtering methods. Still the number of combinations of filtering and grouping methods is too high (24) to display in a single plot. We create a figure for each combination of the color space and kernel function we use for filtering.

It is clear from these figures (and the ones on the previous section) that the grouping methods alone perform much worse than the combinations of filtering and grouping methods. Thus, our claim that *segmentation should be considered as the coupling of a filtering method with a grouping method* is experimentally proved.

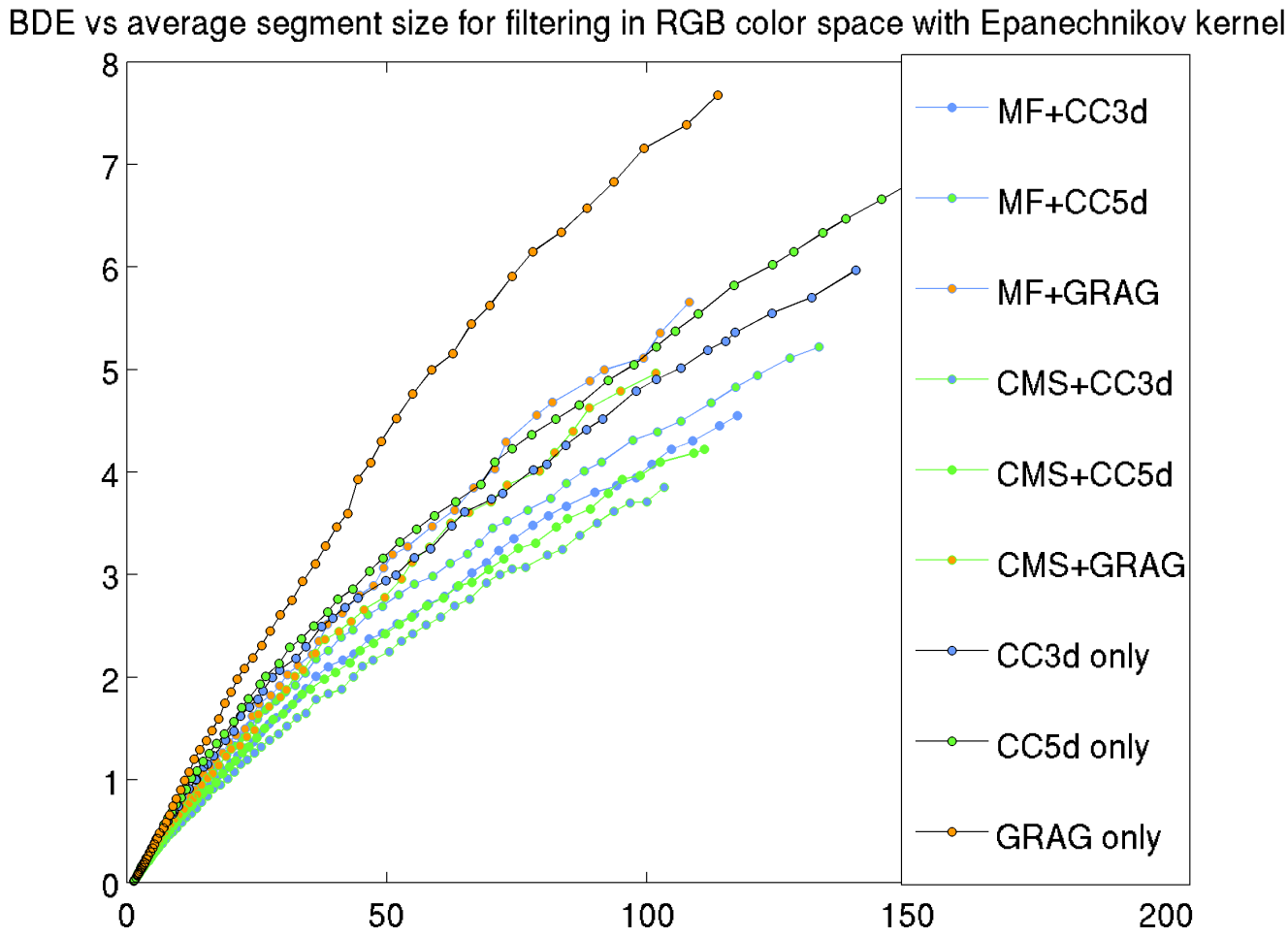


Figure 3.34: BDE vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the RGB color space with an Epanechnikov kernel.

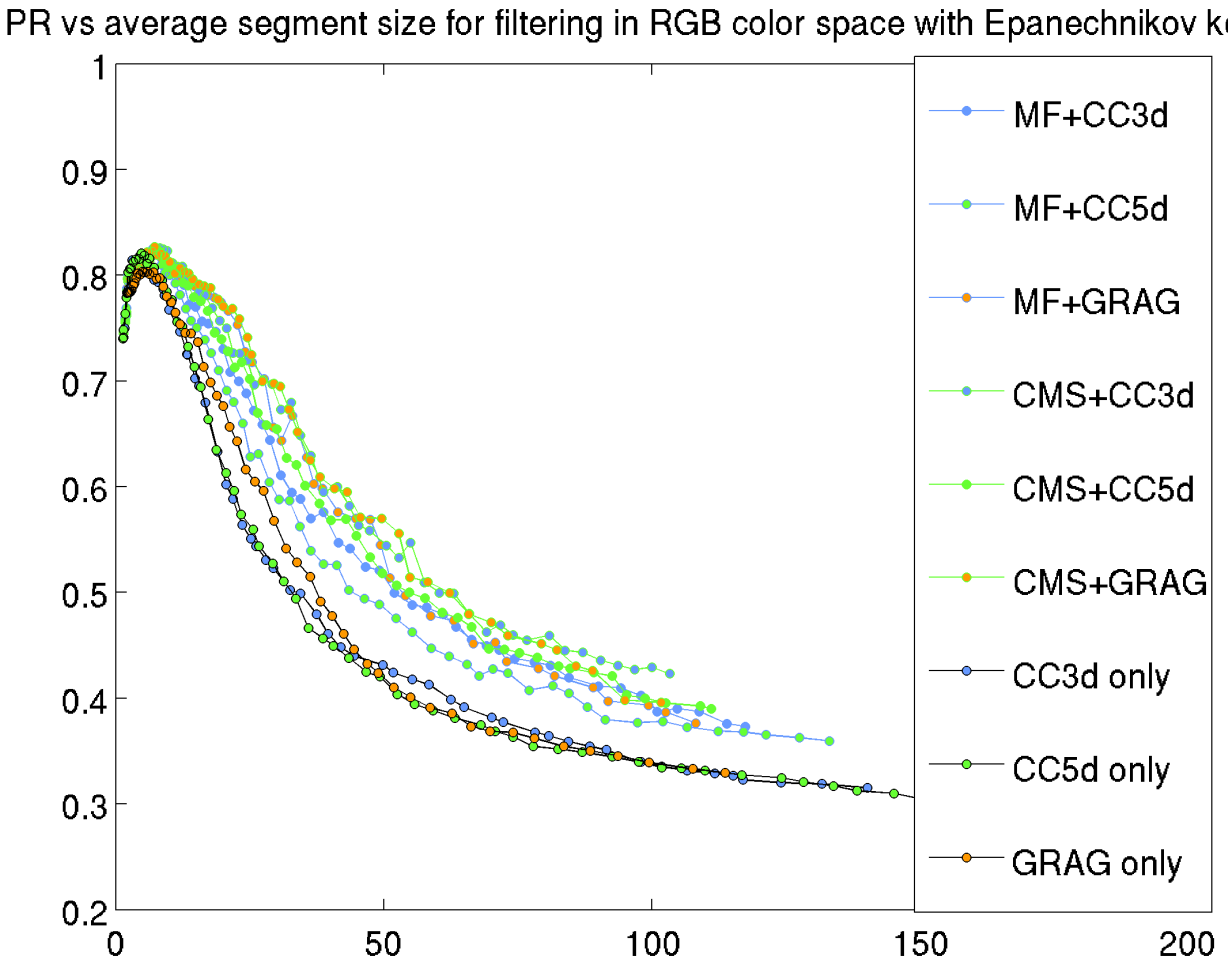


Figure 3.35: PR vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the RGB color space with an Epanechnikov kernel.

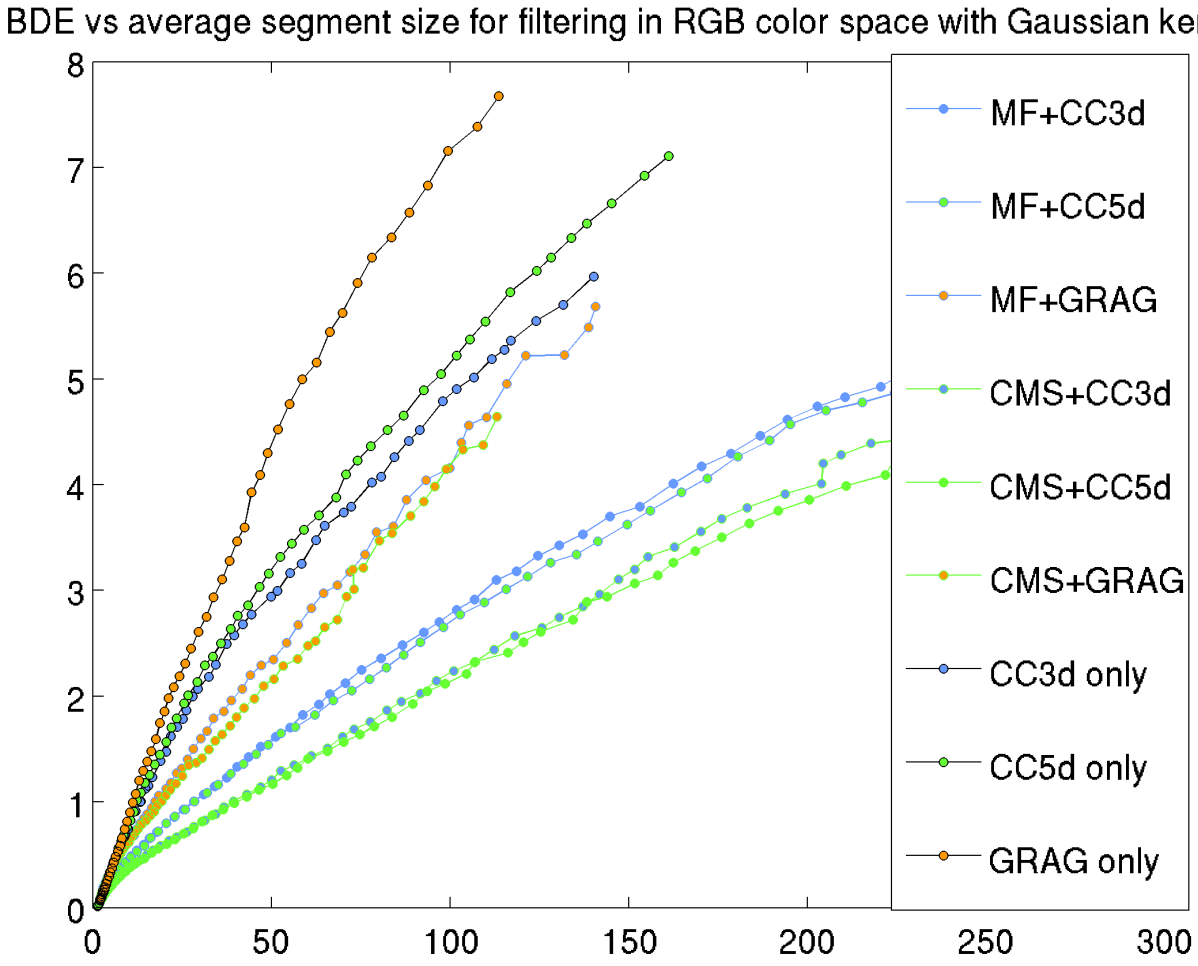


Figure 3.36: BDE vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the RGB color space with a Normal kernel.

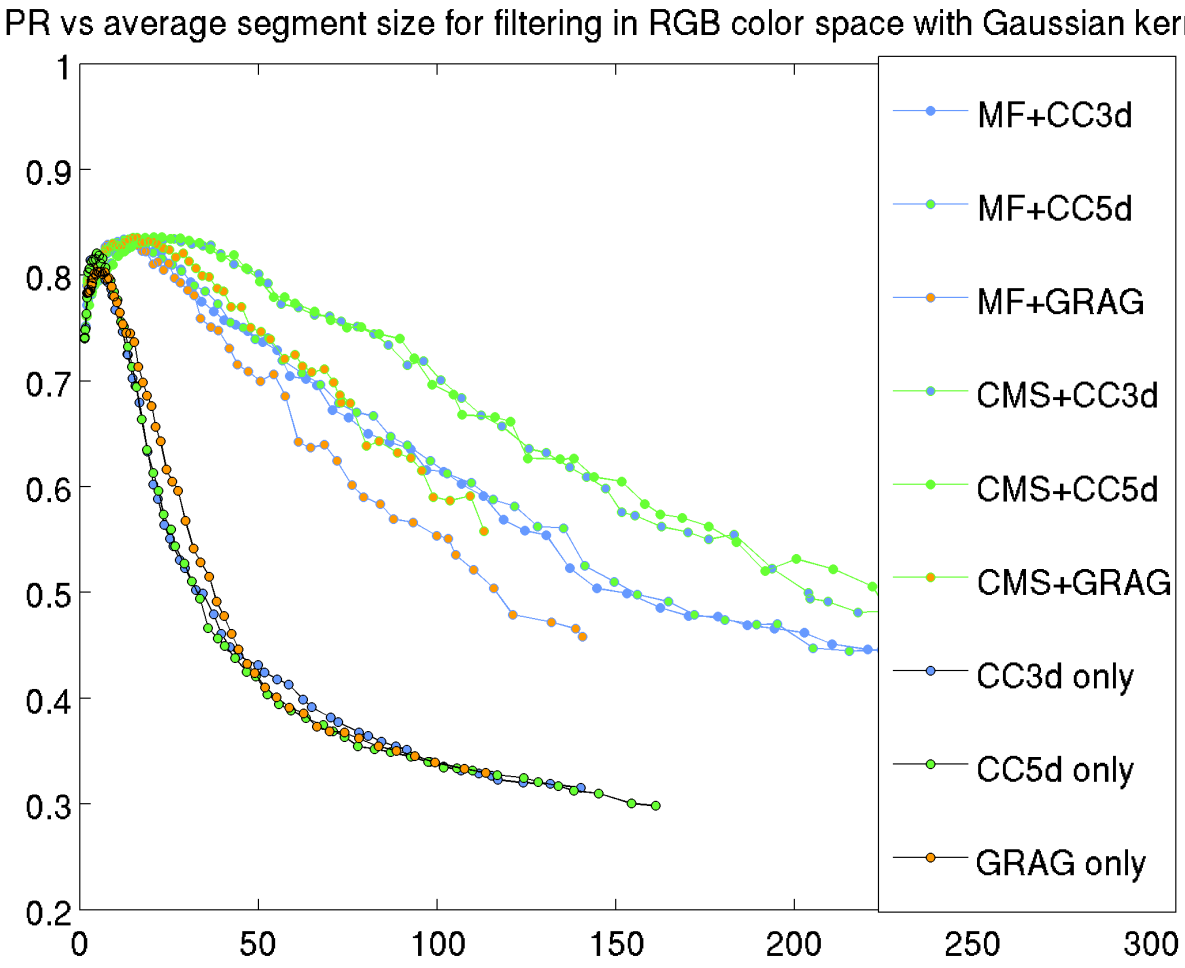


Figure 3.37: PR vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the RGB color space with a Gaussian kernel.

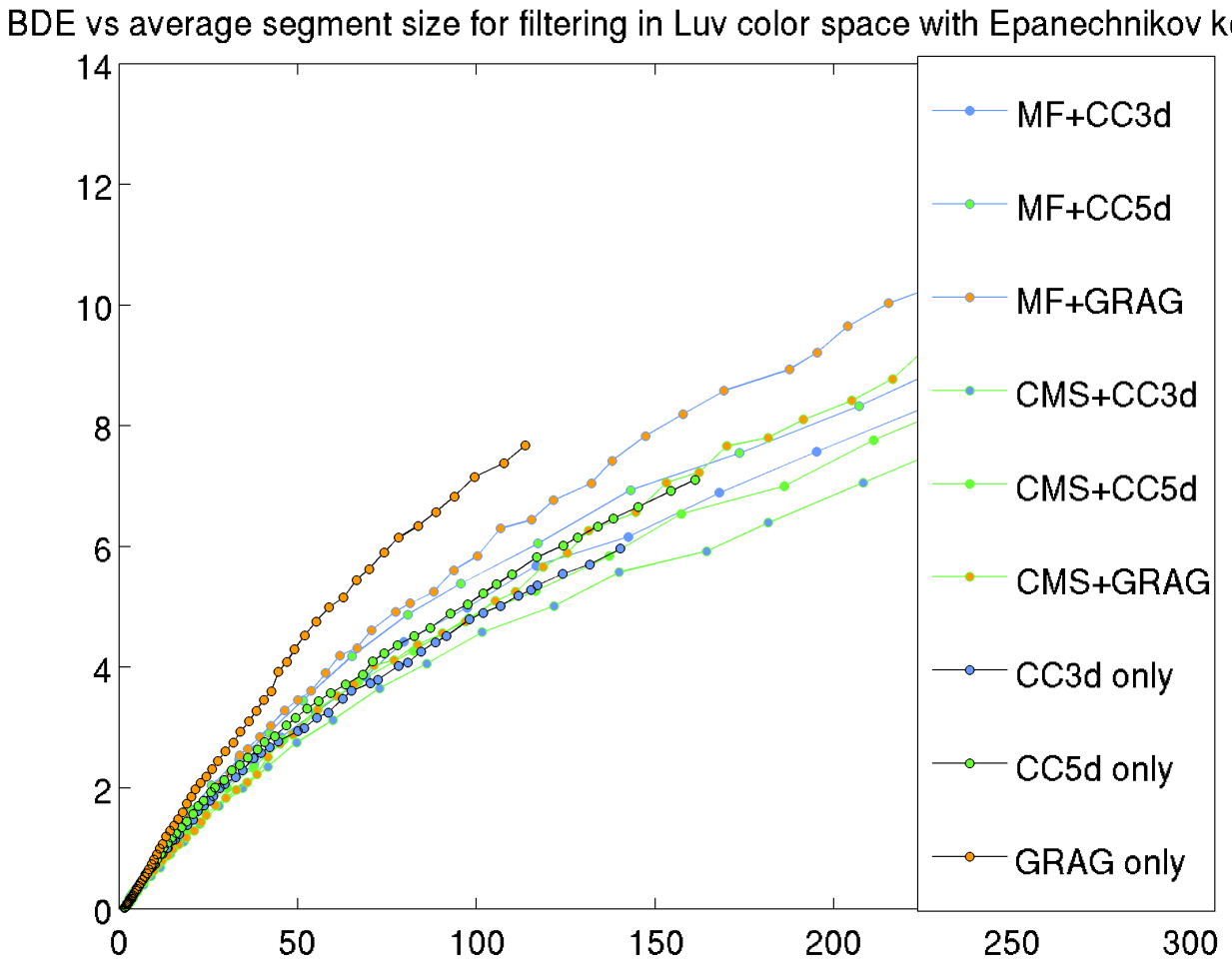


Figure 3.38: BDE vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the Luv color space with an Epanechnikov kernel.

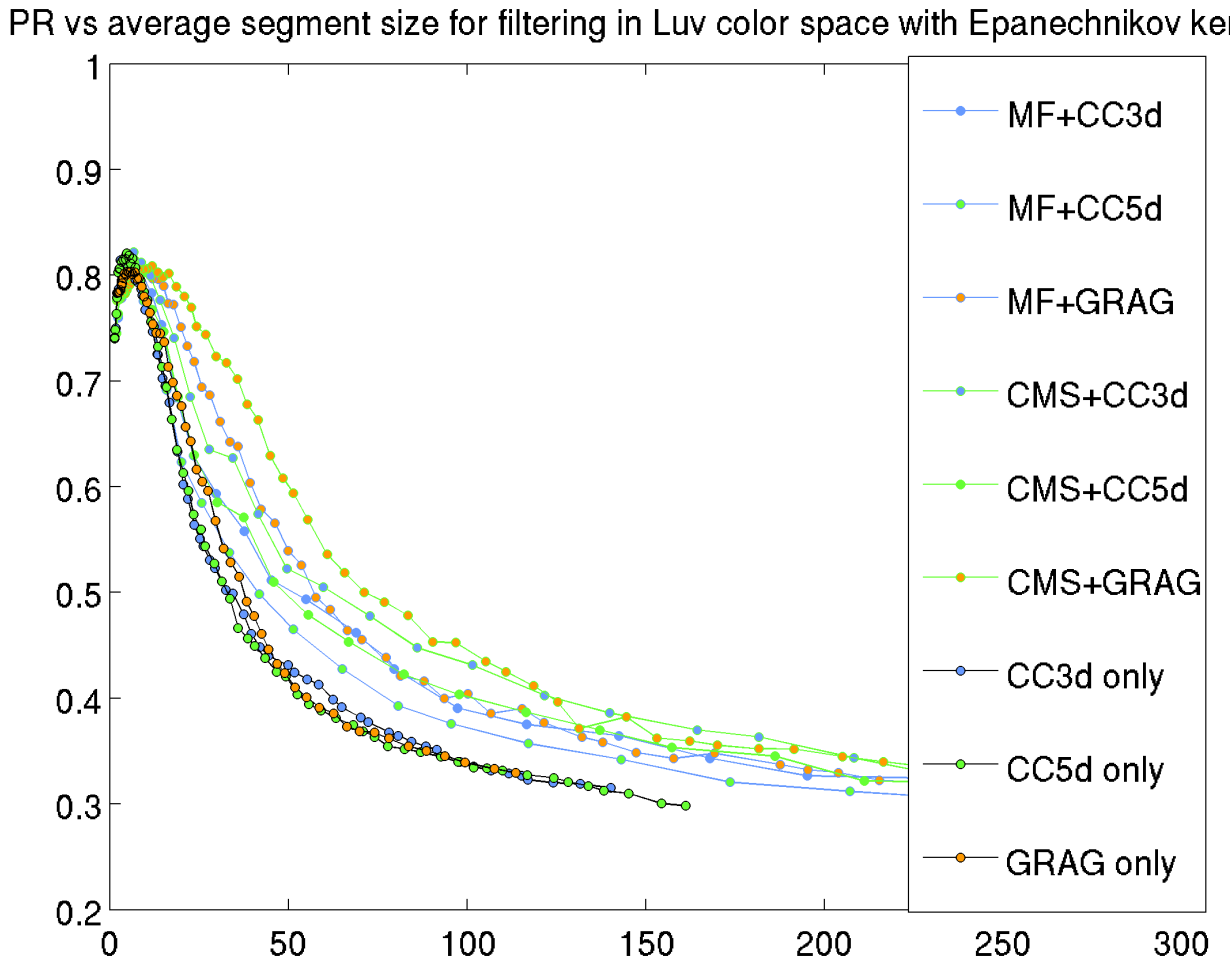


Figure 3.39: PR vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the Luv color space with an Epanechnikov kernel.

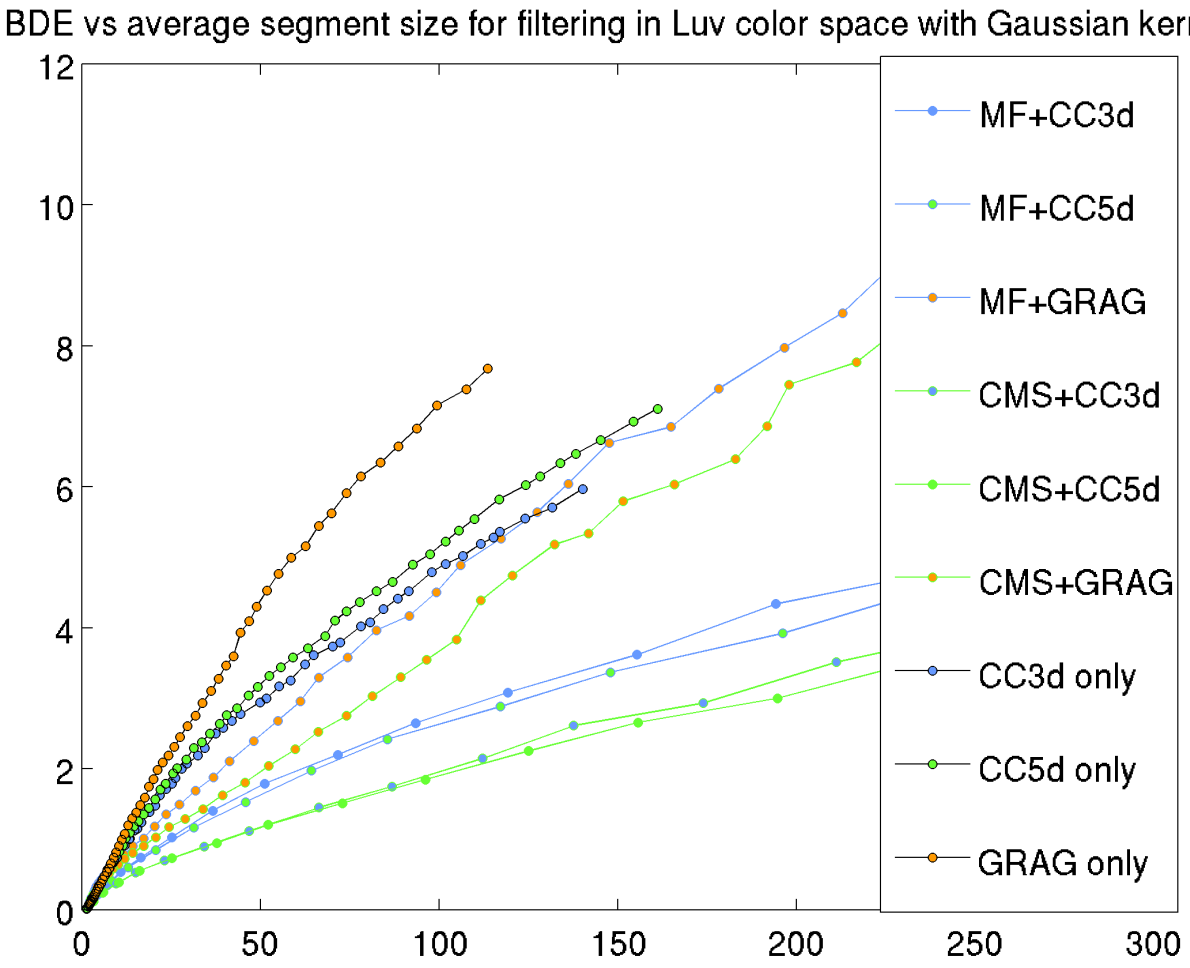


Figure 3.40: BDE vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the Luv color space with a Normal kernel.

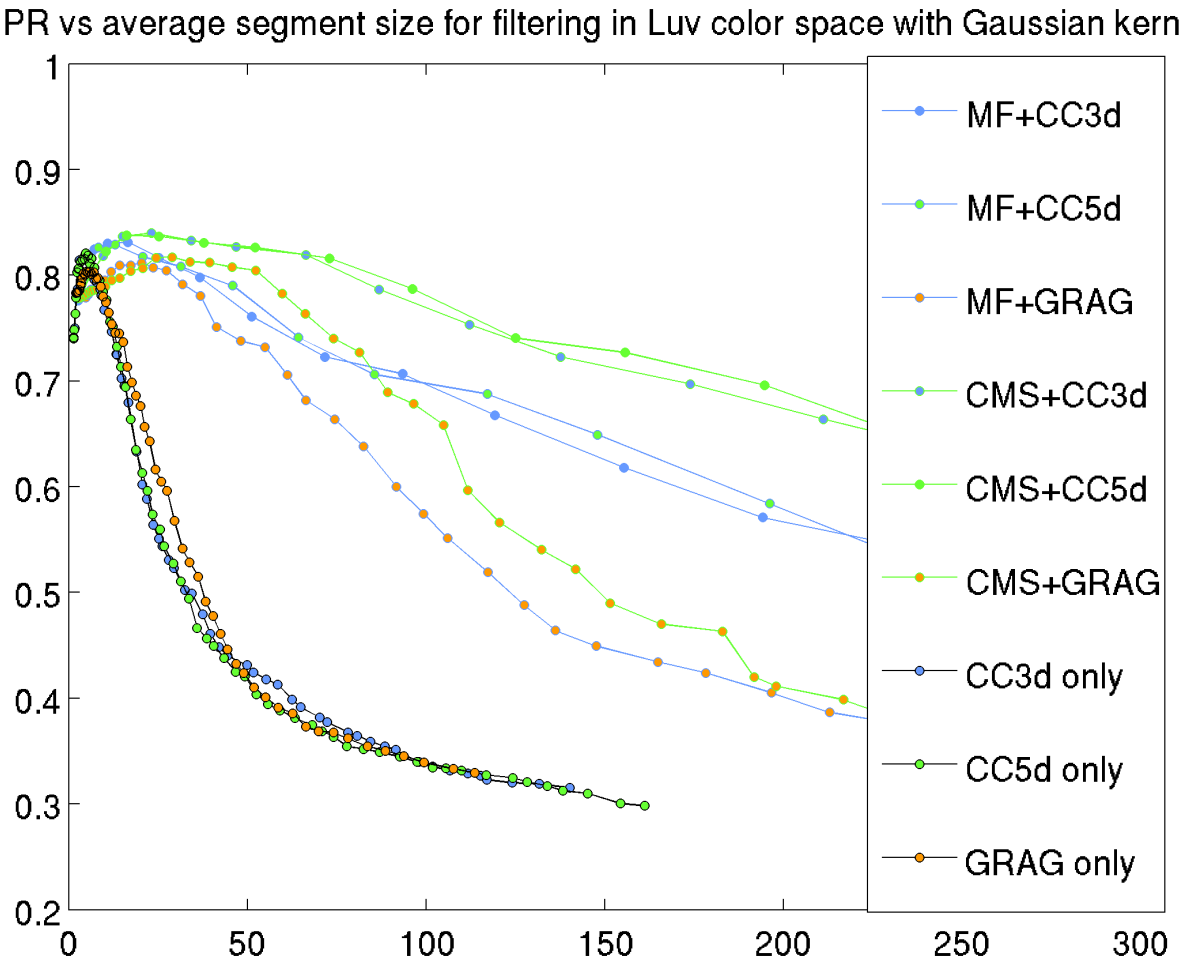


Figure 3.41: PR vs average segment size plots for grouping only methods and filtering+grouping methods. In these plots the filtering is performed in the Luv color space with a Normal kernel.

3.4.5 Compare segmentations for different color spaces and kernel functions

Thus far, almost all the graphs presented the results of various filtering and grouping methods for a specific color space and kernel function. Only Figs. 3.24, 3.25 presented a comparison of a single filtering and grouping method (namely CMS and CC3D) for different color spaces and kernel functions. In this section we try to address the question which color space and kernel function produces the best segmentation results. We only consider three methods (that performed best in the previous experiments), namely MF+GAT with variable k, CMS+CC3D, MF+CC3D.

From Figs 3.42, 3.43 it is clear that the best performing method is the combination of Mode Finding with Grouping with Adaptive Threshold when we use variable k. The next best method is Color Mean Shift with CC3D, while Mode Finding with CC3D performs rather poorly. Furthermore, using the Luv color space seems to be a better option for performing the filtering compared to RGB. Finally, the Normal function produces better results compared to the Epanechnikov kernel function. The difference in the quality of the segmentation (for different color spaces and kernel functions) is not so great in the case of GAT filtering, but it is quite significant when CC3D is used for grouping.

3.4.6 Compare segmentations for different images

In all the previous experiments so far, we presented the cumulative results for the entire database of the 200 images and the 1087 human created segmentations. One

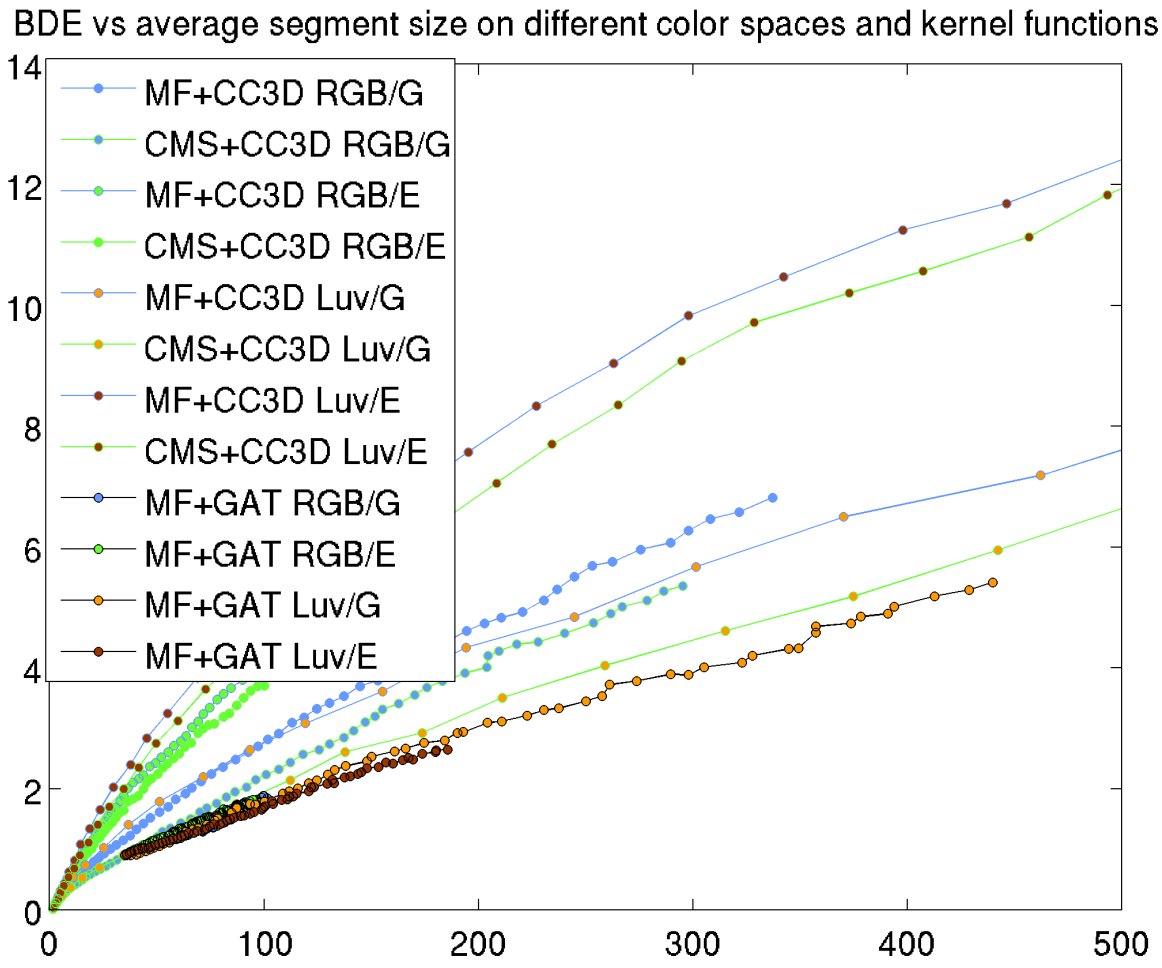


Figure 3.42: BDE vs average segment size plots for three segmentation methods with different color spaces and kernel functions. In the legend "G","E" stand for "Gaussian/Normal" and "Epanechnikov" kernel functions respectively.

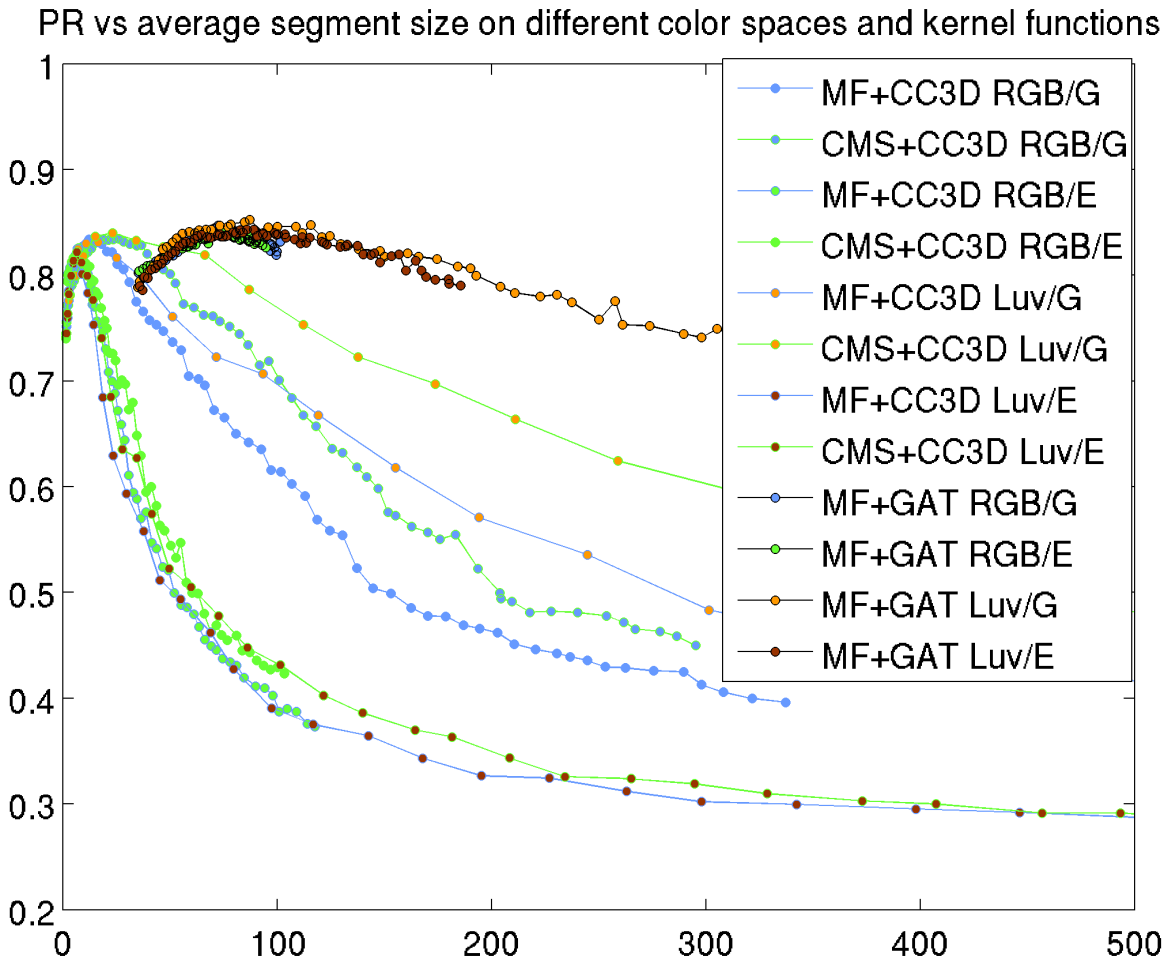


Figure 3.43: PR vs average segment size plots for three segmentation methods with different color spaces and kernel functions. In the legend "G","E" stand for "Gaussian/Normal" and "Epanechnikov" kernel functions respectively.

desired characteristic of any segmentation algorithm is to perform consistently well in a wide range of images. In the previous section we presented the best segmentation algorithms according to the BDE and the PR measures for the whole database of images. In this section we present how these algorithms perform on individual images of this database. For that purpose we display the results on 10 randomly selected images (i.e., 10 segmentations).

The first thing to observe is that MF+GAT is non monotonic on either axis i.e., the average segment size and the comparison measure (BDE or PR) might increase or decrease on the next measurement point. As a consequence the results for all the MF+GAT graphs are quite "chaotic", especially the results when filtering is performed on Luv space with a Normal kernel present a large variation. A careful study of the plots on the different color spaces and kernel functions shows that actually for the same range of average segment sizes filtering on Luv with a Normal kernel is less "chaotic" than the other combinations.

For the other methods (i.e. CMS+CC3D and MF+CC3D) filtering on Luv space with a Normal kernel produces less smooth graphs compared to other color spaces and kernel functions combinations. This is mainly because the results in this combination are good up to a higher average segment value and then they degrade rapidly.

Overall, when segmenting the same image with different segmentation parameters, MF+GAT presents a lower variation in the quality of the segmentation. This means the MF+GAT combination is less sensitive to the selection of the segmentation parameters. CMS+GAT performs slightly better than MF+GAT in the

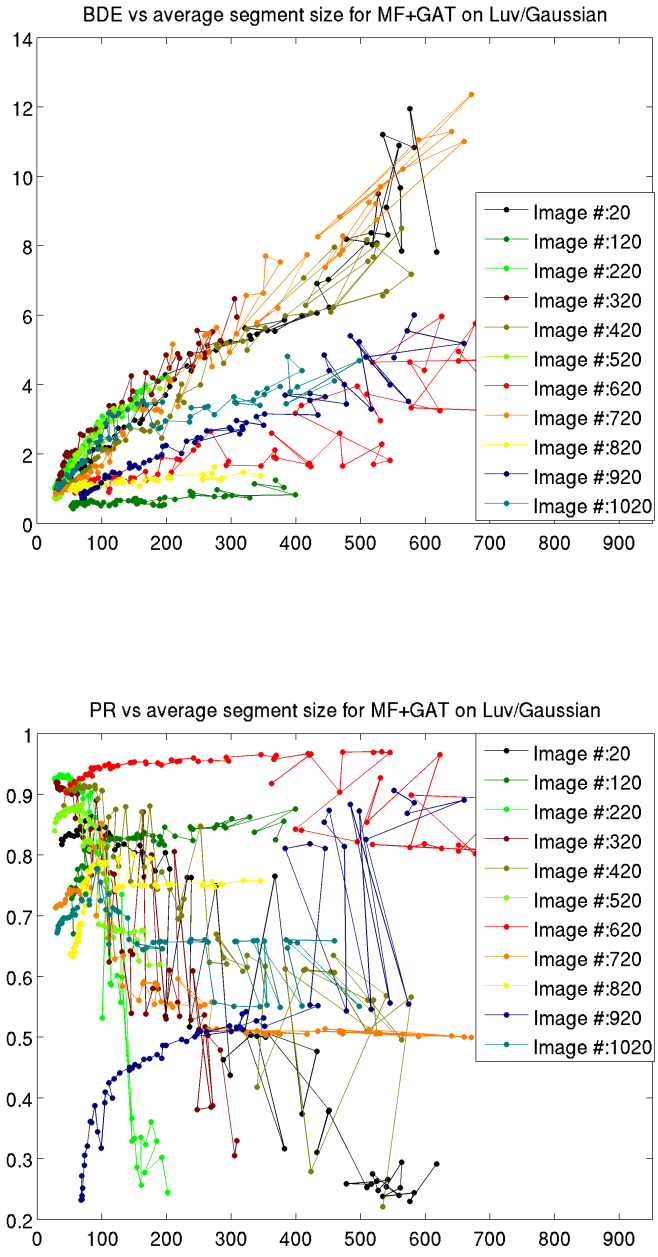


Figure 3.44: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+GAT combination. Filtering is performed in Luv space with a Normal kernel.

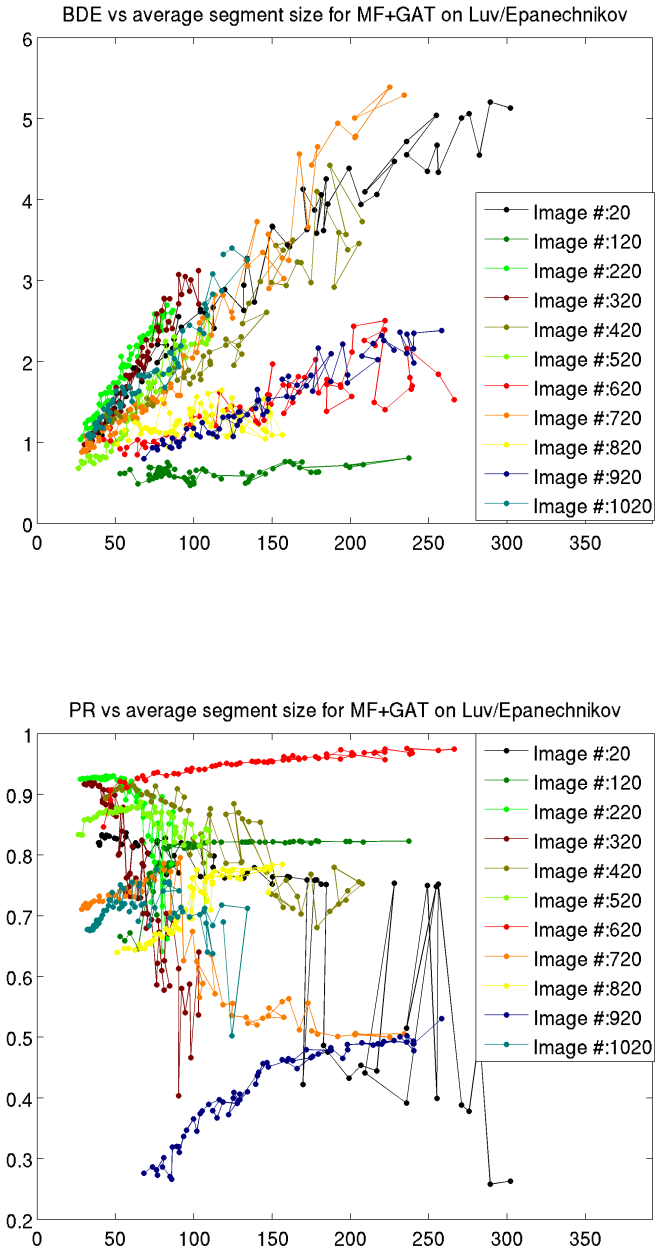


Figure 3.45: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+GAT combination. Filtering is performed in Luv space with an Epanechnikov kernel.

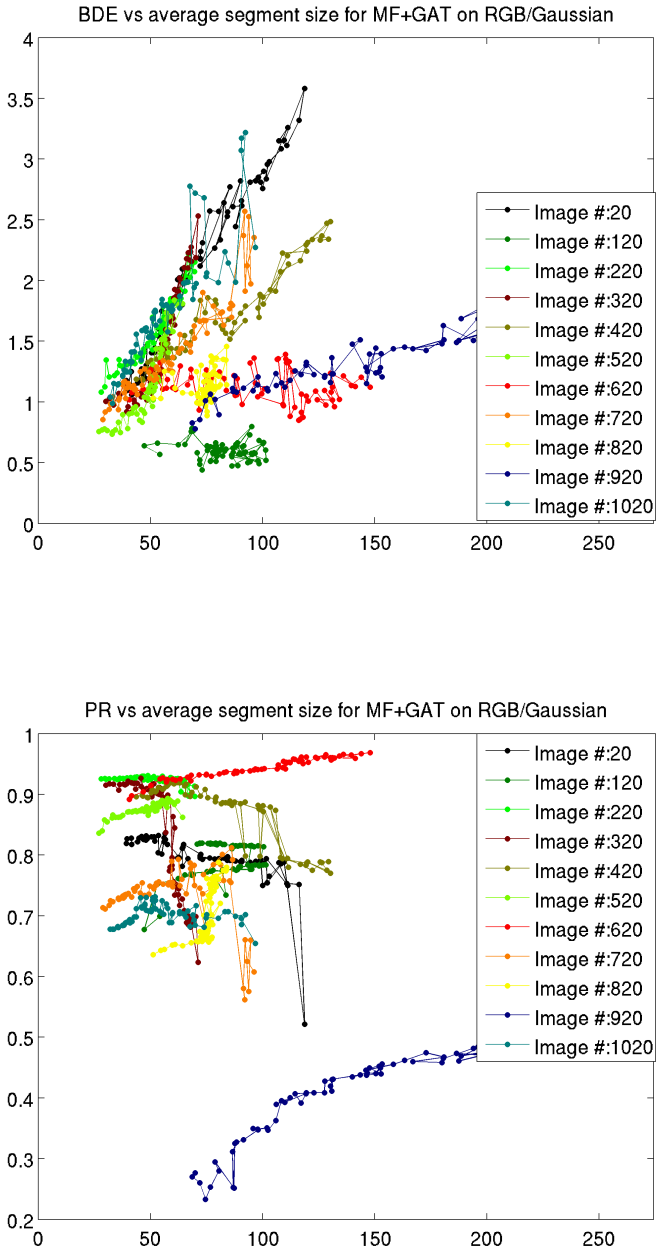


Figure 3.46: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+GAT combination. Filtering is performed in RGB space with a Normal kernel.

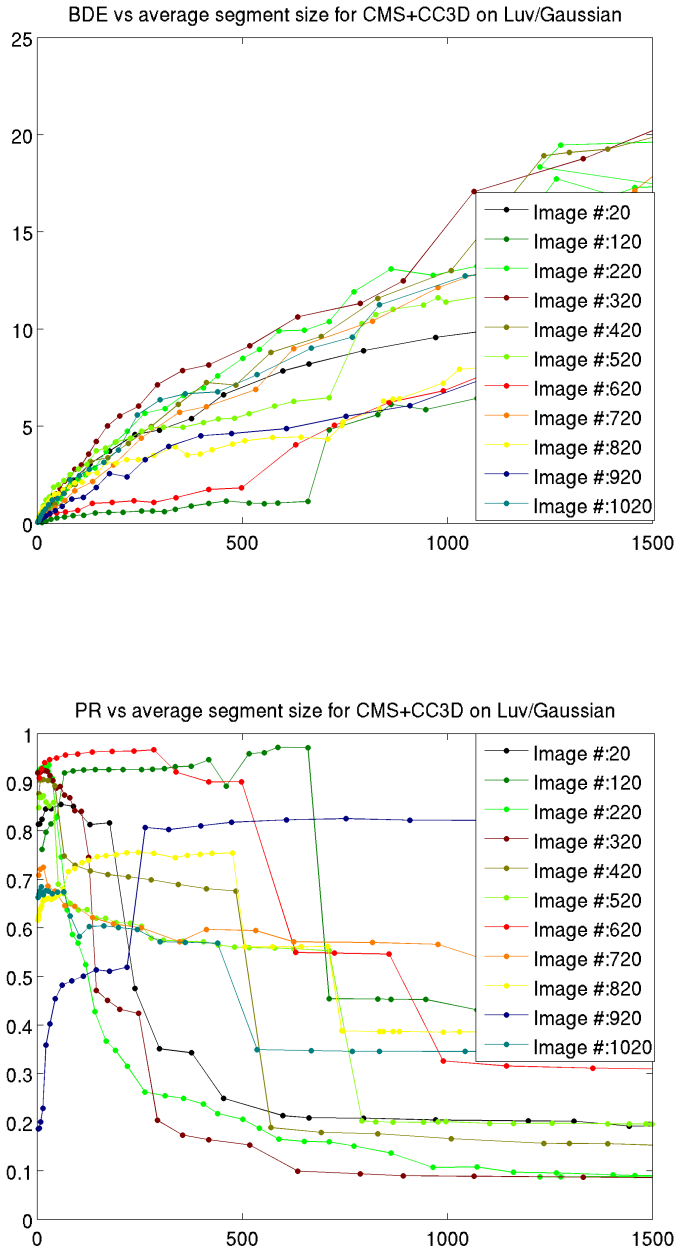


Figure 3.47: BDE and PR vs average segment size plots for individual images of the database segmented with the CMS+CC3D combination. Filtering is performed in Luv space with a Normal kernel.

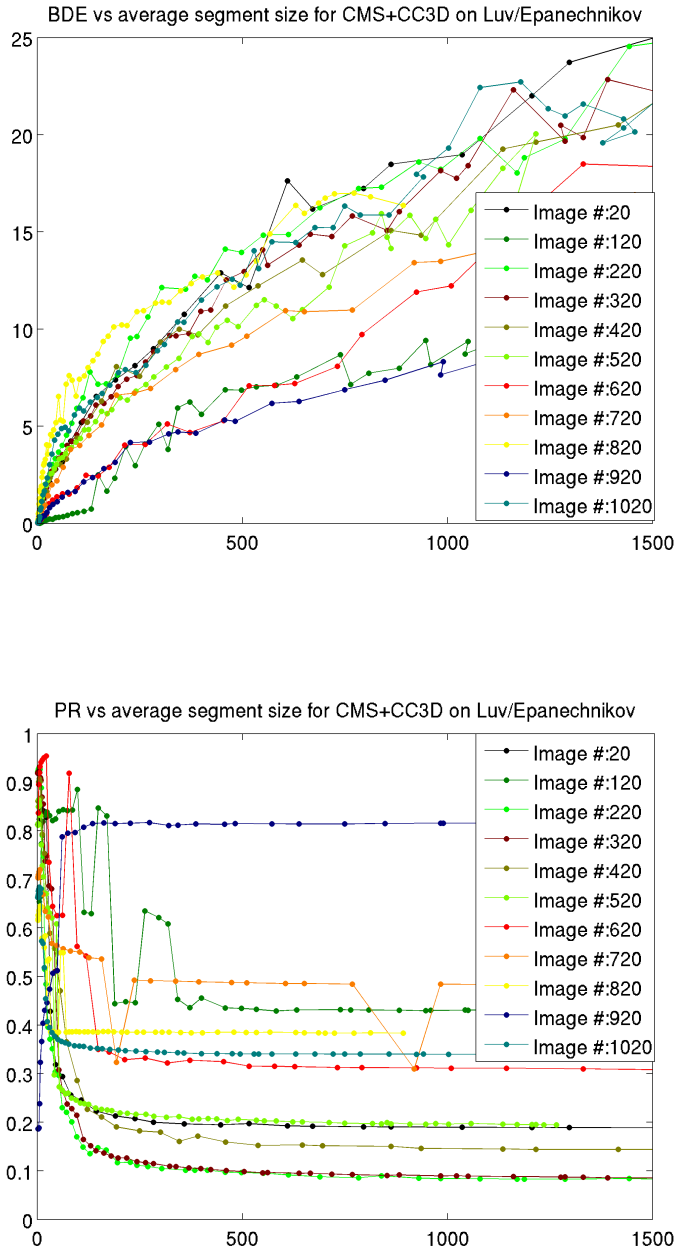


Figure 3.48: BDE and PR vs average segment size plots for individual images of the database segmented with the CMS+CC3D combination. Filtering is performed in Luv space with an Epanechnikov kernel.

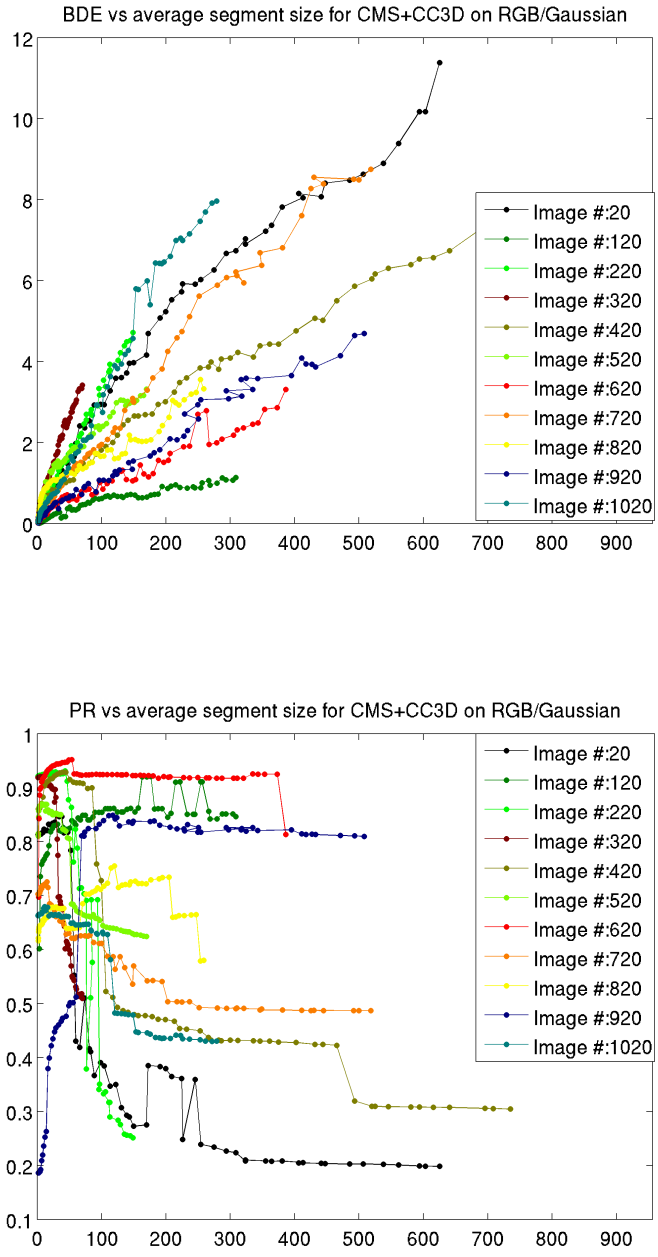


Figure 3.49: BDE and PR vs average segment size plots for individual images of the database segmented with the CMS+CC3D combination. Filtering is performed in RGB space with a Normal kernel.

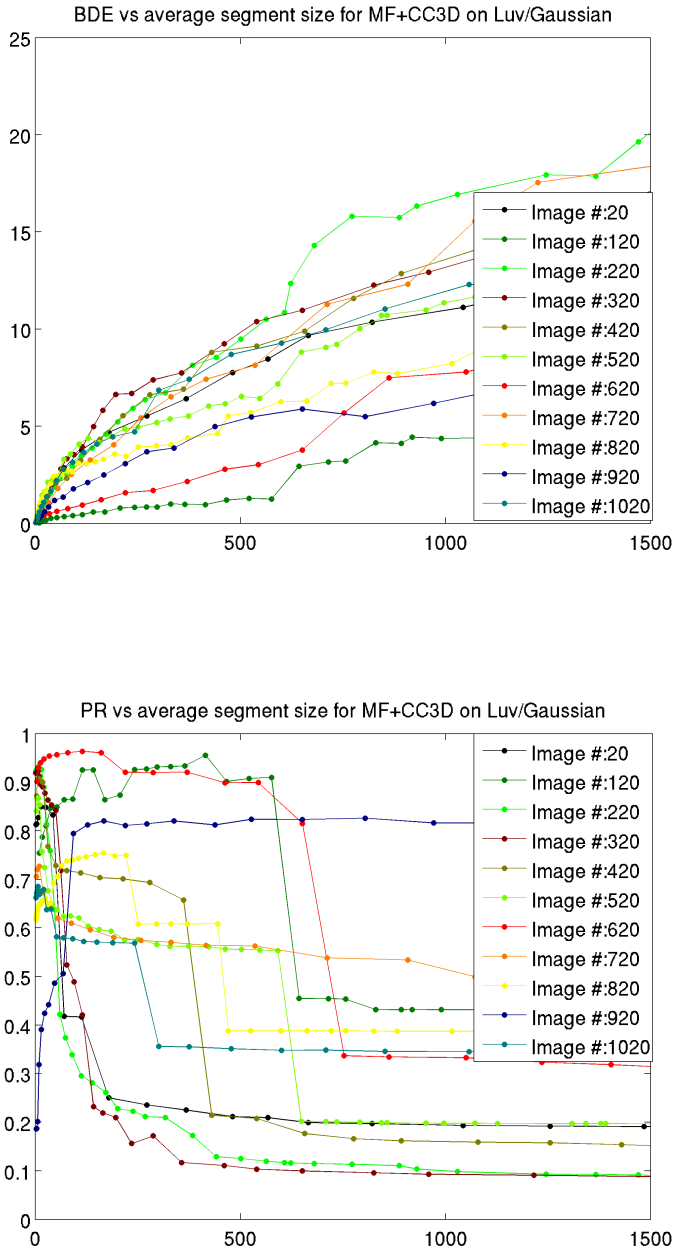


Figure 3.50: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+CC3D combination. Filtering is performed in Luv space with a Normal kernel.

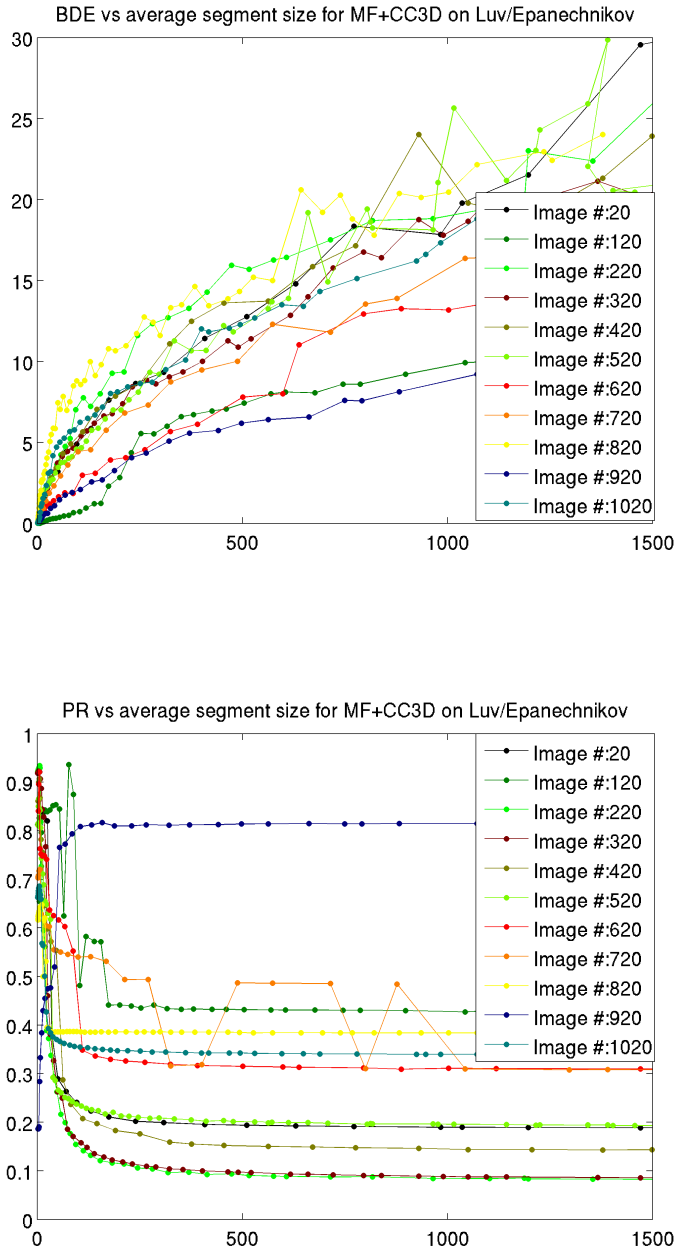


Figure 3.51: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+CC3D combination. Filtering is performed in Luv space with an Epanechnikov kernel.

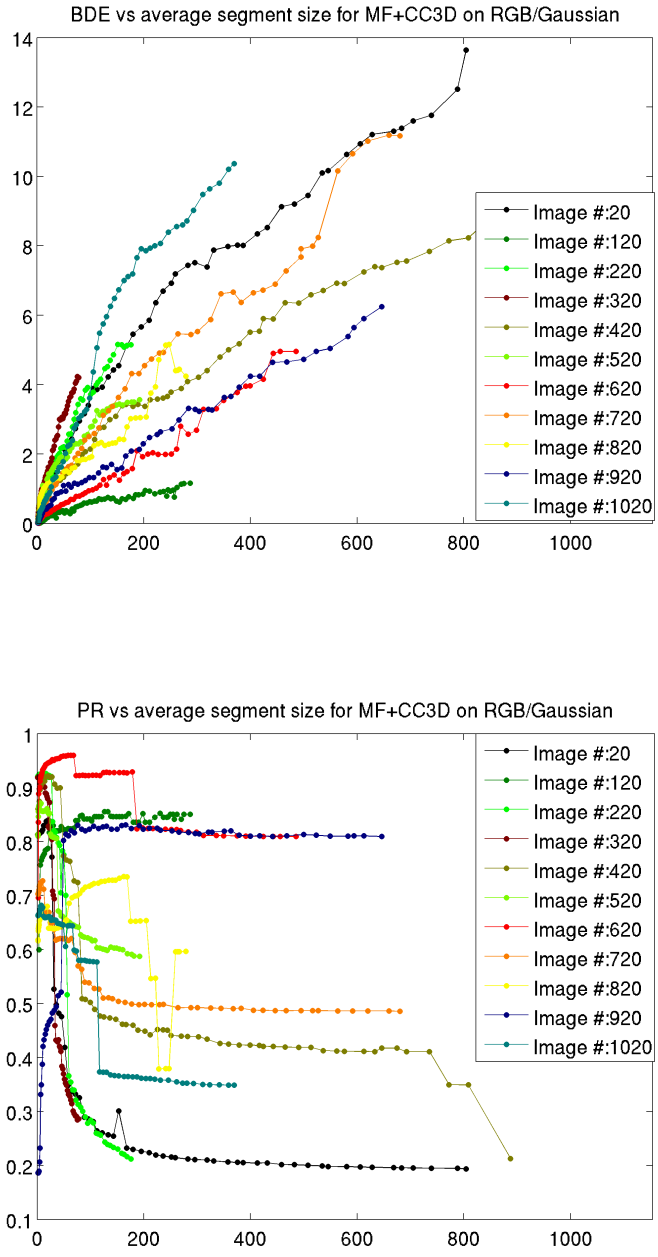


Figure 3.52: BDE and PR vs average segment size plots for individual images of the database segmented with the MF+CC3D combination. Filtering is performed in RGB space with a Normal kernel.

intra-image segmentation quality.

Between different images, MF+GAT also produces the most consistent results in terms of the quality of segmentation. In this case also, CMS+GAT slightly outperforms MF+CC3D.

3.5 Conclusions

In this chapter we presented our position that the problem of color based segmentation should be subdivided into a filtering and a grouping component, and created a number of new segmentation algorithms by combining existing (and new) filtering and grouping methods. We evaluated all the methods extensively, using the Berkeley segmentation dataset and made a number of useful observations. Table 3.2 synopsizes the results of the experimental comparison for performing edge preserving filtering and color based segmentation respectively.

There are two main results that we want to emphasize here. In all the experiments, processing the image with an edge preserving filter before using a grouping method produced significantly better results. Thus it is beneficial to consider *the segmentation process to be a combination of a filtering and a grouping step.*

Second, depending on the grouping method that is used, a different filtering process produces best results. For grouping with a hard threshold (i.e. CC3D, CC5D and GRAG methods) Color Mean Shift filtering worked best. When grouping with an adaptive threshold (i.e. GAT method) Mode Finding proved to be the best method. As a conclusion, *when considering the problem of color based segmentation,*

one should study the combination of the filtering and the grouping method to obtain the best results. Studying only one component in isolation is not sufficient.

Our overall comparison showed that for the Berkeley dataset the best method to use is a combination of Mode Finding with Grouping with Adaptive Threshold (with variable k). Furthermore the results are better when the filtering is performed in Luv color space with a Normal kernel.

There are many interesting directions for future research. Next we present some of them.

As we saw before, the kernel function significantly affects both the filtering and the segmentation results. A more systematic study of this relation, especially why the Normal kernel function produces better results, is an interesting question. Even more so, if one can devise other kernel functions that give even better results. A related question is how one can adjust the kernel function to consider the boundary edge characteristics. Recent work in learning boundary edges (and separating them from texture edges) showed promising results, but it is still an open question how kernel density estimation methods can benefit from such a learning approach.

The previous experiments also proved that different color spaces critically affect the segmentation result. We tested the Luv and RGB color space mainly because these are the color spaces suggested in previous mean shift segmentation papers. This does not exclude the possibility of other color spaces being more beneficial to the segmentation of images. We would be surprised if linear transformations (such as RGB to YUV) would produce significantly different results, but there are unlimited possibilities for non-linear transformations.

Table 3.2: Synopsis of the filtering results

- All segmentation methods are very sensitive to image variations. The methods based on Grouping with an Adaptive Threshold (GAT) are the least sensitive to inter image variation. They also exhibit the least sensitivity to the segmentation parameters (h_r, k) when segmenting the same image.
- Segmentation methods based on GAT grouping are not monotonic.
- Segmentation methods based on GAT grouping outperform , on average, all the other segmentation methods.
- Segmentation methods based on GAT grouping are the most stable to color resolution changes i.e., exhibit less variation of the average segment size.
- Segmentation methods based on CC3D and CC5D grouping have very similar performance, with the CC3D ones producing slightly better segmentation results. The GRAG methods produce better, same or worse depending on the color space and kernel function combination.
- All the graphs of the Global Consistency Error (GCE) measure are misleading because the two segmentations have different number of segments. GCE graphs are misleading since only for a few values for color resolution the number of segments on both segmentations is comparable. That's why we obtain a value close to 0 for very small and very large color resolutions.
- The graphs of the Variation of Information (VI) measure are the least discriminative.
- The graphs of the Probabilistic Rand Index (PR) and Boundary Displacement Error (BDE) measures are the most discriminative.
- Segmentations obtained by grouping methods alone have much lower quality than the ones obtained using a combination of a filtering and a grouping method.
- Color Mean Shift (CMS) based segmentation methods outperform all the other filtering methods when they are combined with CC3D or CC5D or GRAG grouping methods.
- When using GAT grouping with varying parameter k Mode Finding (MF) produces the best results.
- Filtering in Luv produces much larger segments than filtering in RGB for a given color resolution h_r. Filtering with a Normal kernel results in larger segments compared to using a Epanechnikov kernel.
- The selection of the kernel function seems to be very important for the segmentation results. More specifically, we obtained the best segmentation results when the filtering was performed with a Normal kernel in the Luv color space. The second best configuration is a Normal kernel with an RGB color space, while the results obtained with an Epanechnikov kernel in either RGB or Luv color spaces are much worse.

In this thesis we mostly focused on the filtering part of the segmentation process. For the grouping part we selected a few, simple and fast methods. In the computer vision literature there is a large variety of methods that are used for image clustering. Energy minimization methods (e.g. graph cuts), eigenvector based methods (e.g. normalized cuts) and soft assignment methods based on algebraic multigrid are also legitimate candidates for the grouping part. It is interesting to see the quality of the segmentation using these clustering methods.

Further study is required on the optimal combination of the filtering parameters (namely the color resolution h_r) with the segmentation parameters (e.g. in the case of GAT k). Their relation that produces the best segmentation results for different image sizes is yet to be determined.

Finally, in all the experiments we use the implicit plot of the quality measure over the average segment size as an indication for the quality of the segmentation. Our goal is to use color segmentation to generate hypotheses for planar surfaces and as such the larger the segment the better we can verify whether it has a consistent surface normal or not. A wide variety of applications exist that use color segmentation as a first step and in some of them other characteristics (than segment size) might be more important. For example, stereo methods are more worried whether a segment crosses occlusion boundaries or not. It would be interesting to see how the segmentation algorithms that we presented above fare under different measures.

Chapter 4

Combining Cues for Surface Normal Estimation

4.1 Introduction

In this chapter we switch our attention from the problem of color based segmentation to the problem of surface normal estimation. It is widely accepted that changes (over multiple frames) on the boundaries and the texture of an image region provide complimentary information about the shape and the $3D$ position of the corresponding object. Thus, combining methods based on boundary extraction with ones on textured regions results in more robust and accurate estimation. Especially, for relatively simple environments, such as corridors, it is often the case that only one type of cue will be present and thus only one type of method will provide reliable measurements. Furthermore, in such environments the predominant shape of objects is planar and the object boundaries are usually lines.

Motivated by the above observations, this chapter proposes two methods to estimate the $3D$ position of planar objects; the first considers the change of the texture and the second the change of image lines. More specifically, the main contributions of the chapter are:

- We present a novel image line constraint for estimating the 3D orientation of planes (Sec. 4.3).

- We describe a novel technique to compute the $3D$ shape from the change of texture for planar objects based on harmonic analysis (Sec. 4.4).

- We present experimental results on how accurate the two methods perform in real indoor environments. The integration of the two methods with the odometry readings from the robot's wheels using an extended Kalman filter, outperforms the results obtained by each method in isolation (Sec. 4.6).

- We experimentally show that the proposed method allows for navigation in environments where little texture is present using a simple motion control policy (Sec. 4.8).

4.1.1 Related Work

The computer vision community has long studied the structure from motion (SfM) problem ([43],[44]) and recently focused on large-scale 3D reconstruction (e.g. [45]). Following the success of Simultaneous Localization and Mapping (SLAM) using range (especially laser) sensors ([46]), the robotics community has migrated the existing methods to work with data from cameras. Usually, the environment is represented with a set of image feature points, whose pose is tracked over multiple frames ([47]). Often, image features are more informative than range data, but the estimation of their 3D position is much less accurate. Straight lines are common in man-made environments and are arguably more reliable features than points, thus they have been used before in structure from motion ([48], [49]) and SLAM ([50]). Our method is about computing 3D structure information in a simplified SfM

situation, but very robustly. We use a formulation of line constraints that separates slant from distance estimation. Thus, it is different from the ones classically used in SfM.

On the other end of the spectrum there are methods belonging to the *mapless visual navigation* category ([51]), where no prior knowledge about the environment is assumed and no spatial representation of it, is created. Most of that work is inspired by biological systems. A survey of such methods implementing the centering behavior can be found in [52]. More specifically, systems capable of avoiding walls and navigating in indoors environments using direct flow-based visual information obtained from a single wide-FOV camera facing forwards ([53], [54], [55]), multiple cameras facing sideways ([56], [57]) or panoramic cameras ([58]), have been implemented. Our approach is also different from the aforementioned, because we first estimate an intermediate state of the environment (in terms of surface normals) and we use this for navigation.

The general method for estimating the stretch and shift of a signal using the log of the magnitude of the Fourier transform, known as *Cepstral analysis*, was first introduced by Bogert et al. [59] and was made widely known by Oppenheim and Schafer [60]. It is commonly used in speech processing [61] to separate different parts of the speech signal.

Frequency based techniques exploiting the phase shift theorem have been used in computer vision for image registration (in conjunction with the log-polar transform of an image), e.g. [62], [63], [64] and optical flow computation ([65]). Phase correlation, however, has not been used for shape estimation.

4.2 Problem Statement and terminology

Due to the completely different topic of this chapter we need to redefine our notation and terminology. Hence, in this section we introduce some common symbols that are used in the rest of the chapter and present the problem that we tackle in the following three sections. For simplicity and improved readability reasons, all the equations in Sec. 4.3,4.4 and 4.5 are expressed in the camera coordinate system (where the images were acquired). In Sec. 4.6 and 4.7 we transfer the estimates in the robot-centric coordinate system (Fig. 4.5). Vectors are denoted with an overhead arrow and matrices with bold letters.

We denote with $\overrightarrow{T}$, $\mathbf{R}$ the translation and rotation between two frames respectively, with $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$ a plane in the 3D world and with $\overrightarrow{n} = \frac{\overrightarrow{N}}{|\overrightarrow{N}|}$, the plane normal. Also $\overrightarrow{P} = (X, Y, Z)^T$ is a 3D point. When $\overrightarrow{P}$ belongs to $\overrightarrow{N}$ then $\overrightarrow{P} \cdot \overrightarrow{N} = 1 \Leftrightarrow \alpha X + \beta Y + \gamma Z = 1$. The image plane is assumed to lie on the plane $\mathbb{I} : Z = f$, where f is the focal length of the camera. Then, the projection of $\overrightarrow{P}$ on $\mathbb{I}$ is $\overrightarrow{p} = (x, y, f)^T = \frac{f}{Z}(X, Y, Z)^T$. The inverse depth at $\overrightarrow{P}$ amounts to

$$\frac{1}{Z} = \alpha \frac{x}{f} + \beta \frac{y}{f} + \gamma \tag{4.1}$$

Given the translation and rotation of the camera between two images we seek to *estimate the plane parameters* $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$.

4.3 Orientation and Distance from lines

Here we describe a constraint for recovering the orientation of a world plane from image lines. The constraint can be used in two ways: first as a multiple view constraint, where we use the images of a single line in 3D in two views [66]; second as a single view constraint where we use the images of two parallel lines in 3D in one view.

4.3.1 Single Line in Multiple Frames

As shown in Fig. 4.1, consider two views with camera centers O_1 and O_2, which are related by a rotation $\mathbf{R}$ and a translation $\overrightarrow{T}$. A 3D line $\mathcal{L}$ lies on the plane with surface normal $\vec{n} = \frac{\vec{N}}{|N|}$. $\mathcal{L}$ is projected in the two views as l_1 and l_2. Let $\vec{l_{m1}}$ be the representation of l_1 in the first camera coordinate system as a unit vector perpendicular to the plane through $\mathcal{L}$ and O_1. Similarly, let $\vec{l_{m2}}$ be the representation of l_2 in the second camera coordinate system as a unit vector perpendicular to the plane through $\mathcal{L}$ and O_2. The two planes perpendicular to $\vec{l_{m1}}$ and $\vec{l_{m2}}$ intersect in $\mathcal{L}$[1]. Expressing this relation in the first camera coordinate system, we have

$$\mathcal{L} \parallel \vec{l_{m1}} \times \mathbf{R}^T \vec{l_{m2}}, \tag{4.2}$$

and since $\vec{n}$ is perpendicular to $\mathcal{L}$, we have

[1] The necessary and sufficient condition for the two planes to be different is that the translation $\overrightarrow{T}$ is not parallel to the line $\mathcal{L}$.

$$(\vec{l_{m1}} \times \mathbf{R}^T \vec{l_{m2}}) \cdot \vec{n} = 0. \tag{4.3}$$

Practically, we want to avoid computing the correspondence of two lines in two frames, so we adopt the continuous representation of Eq. 4.3 as

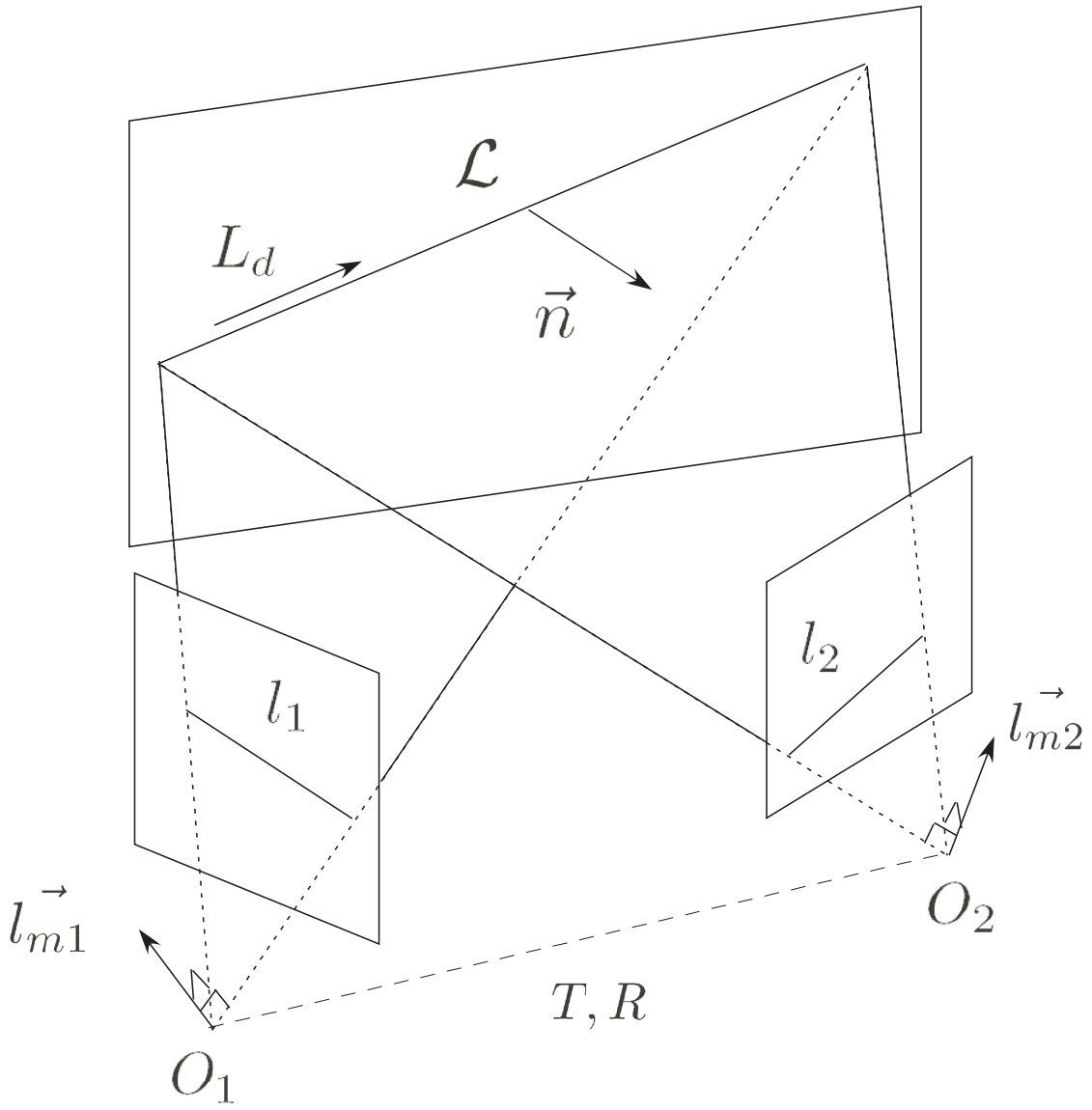
$$(l_1 \times (\dot{l_1} - \vec{\omega} \times l_1)) \cdot \vec{n} = 0, \tag{4.4}$$

where l_1 denotes l_{m1}, $\vec{\omega}$ is the angular velocity of the robot and $\dot{l_1}$ is the temporal derivative of the line that can be computed from the normal flow.

This is the linear equation we use to estimate $\vec{n}$. Notice, this constraint (which intuitively is known as orientation disparity in visual psychology) allows us to estimate the surface normal (that is the shape) of the plane in view, using only rotation information. At this point we should also note that no distance information is encoded to vector $\vec{n}$, which is of unit length.

4.3.2 Two or More Lines in the Same Frame

We can use the constraint in Eq. 4.4 also from one view. Imagine that two views are related by a translation only, or similarly consider two parallel lines in one view. Given two lines l_1 and l_2 that are projected from two parallel lines, $\mathcal{L}_1$ and $\mathcal{L}_2$, in the 3D scene, we recover the orientation of $\mathcal{L}_1$ and $\mathcal{L}_2$ using Eq. 4.2 (Fig. 4.2). Assuming $\mathcal{L}_1$ and $\mathcal{L}_2$ lie on the same wall, which is perpendicular to the ground, and $\vec{n} = \frac{\vec{N}}{|N|}$ as its surface normal, we then recover the surface normal of the wall



(a) Line constraint in multiple views

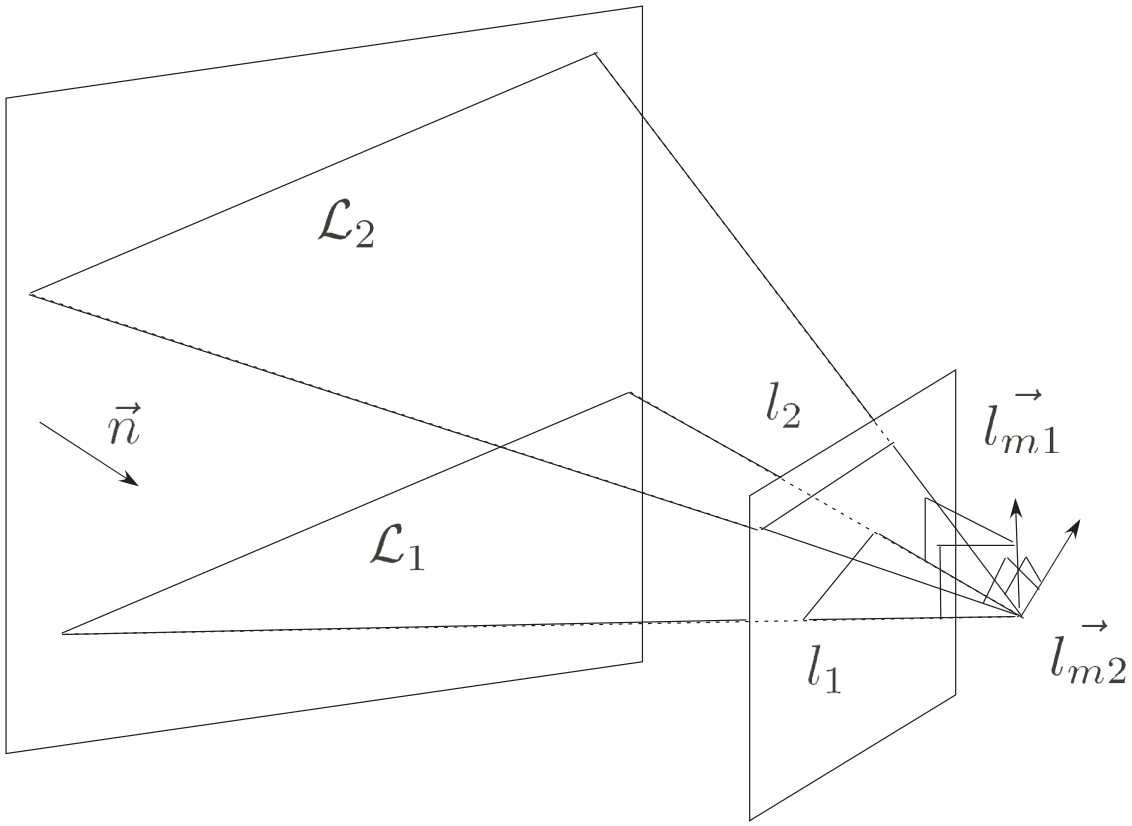
Figure 4.1: A single line is projected to two images from different viewpoints.

from

$$(\vec{l_{m1}} \times \vec{l_{m2}}) \cdot \vec{n} = 0. \tag{4.5}$$

If we have more than two lines that are generated by parallel 3D lines, we can average results from Eq. 4.5.

The constraints discussed above provide better information than vanishing point. From two or more 3D lines, a general plane can be reconstructed. In our case, the plane is perpendicular to the ground plane, thus the surface normal can be described by only one parameter, i.e. $\frac{\alpha}{\gamma}$ (because $\vec{N} = (\alpha, 0, \gamma)^T$). In general, the robot can move based on the position with respect to the line.



(a) Line constraint in a single view

Figure 4.2: Two $3D$ lines, belonging to the same plane, are projected to two image lines.

4.3.3 Distance estimation

After we have computed the slant of the plane, we can also estimate its distance. For this we need the translation T. The distance $d_{\mathcal{L}}$ of the line $\mathcal{L}$ from the camera amounts to [67]

$$d_{\mathcal{L}} = \frac{(l_1 \cdot \overrightarrow{T})}{(\dot{l}_1 + (l_1 \times \vec{\omega}))^T (l_1 \times \overrightarrow{L}_d)}, \tag{4.6}$$

with $\overrightarrow{L}_d$ a unit vector parallel to $\mathcal{L}$, computed as

$$\overrightarrow{L}_d = \frac{l_1 \times (\dot{l}_1 + l_1 \times \vec{\omega})}{|l_1 \times (\dot{l}_1 + l_1 \times \omega)|} \tag{4.7}$$

and the distance d of the plane from the camera is computed as

$$d = d_{\mathcal{L}} \overrightarrow{n} \cdot (l_1 \times \overrightarrow{L}_d) \tag{4.8}$$

4.3.4 Implementation details

To obtain accurate measurements of lines, we modified P. Kovesi's Matlab code[2]. The unoptimized Matlab version of the slant estimation code based on lines runs in ~ 1.5 seconds per iteration on our test bed (a 1.5 GHz Pentium M laptop with 768MB RAM).

In Fig. 4.3 we present three representative frames obtained from the front camera. Note that we did not introduce any artificial landmarks, thus only objects existing in the environment, like doors and door frames are present. To find "good"

[2]http://www.csse.uwa.edu.au/~pk/research/matlabfns

lines to track, we further assume that the longest lines present in the scene are the ones on the boundary between the floor and the walls. Thus, using a threshold on the line length we are able to remove all other lines. In Figs. 4.6 and 4.7 we present the distance and slant estimates which we obtained using the line constraint for a test sequence of 20 frames. We observe that the slant is estimated with good accuracy, while the distance estimation is not very accurate.

4.4 Harmonic shape from texture for planar surfaces

4.4.1 Theory

In this section we assume that the camera is parallel, and the wall perpendicular to the ground. Thus $\overrightarrow{N}$ further simplifies to $(\alpha, 0, \gamma)^T$ and Eq. 4.1 becomes

$$\frac{1}{Z} = \alpha \frac{x}{f} + \gamma \tag{4.9}$$

Consider that we acquire two images I_1 and I_2 and that we know (from the odometry readings) the translation $\overrightarrow{T} = (T_x, 0, T_z)^T$ and rotation $\mathbf{R}$ relating I_1 and I_2. The

(a) First image (b) Second image (c) Third image

Figure 4.3: Three frames of our line testing sequence, with the detected lines drawn in yellow color. In all cases the lines are well localized.

Algorithm 4.1 Match Epipolar Lines

Input:

p : Image point in first image

T, R : Translation/Rotation

K : Camera matrix

D : Reference distance, randomly chosen

Output:

$[p_1, p_2]$: Set of corresponding points in first and second image along the epipolar lines

Algorithm:

Compute Essential Matrix : $E = [T]_x R$

Compute Fundamental Matrix : $F = K^{-T} E K^{-1}$

Compute Epipolar Line in Second Image : $l_2 = Fp$

Compute Corresponding Epipolar line in first image using D

first step is to locate corresponding epipolar lines on the two images (Fig. 4.4) using the procedure described in Alg. 4.1.

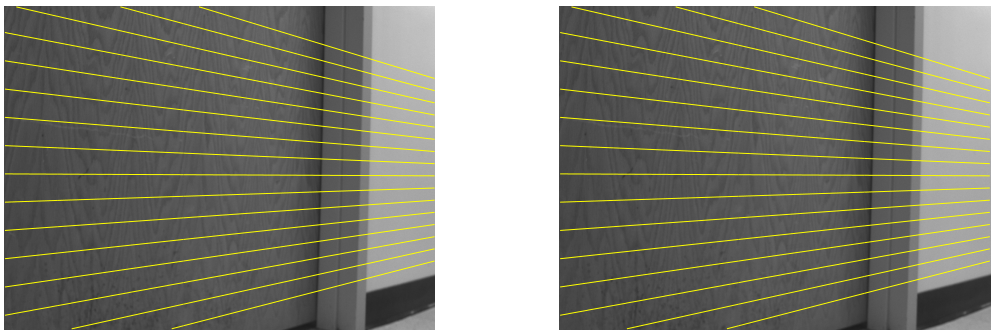


Figure 4.4: The epipolar lines for two frames. The translation vector is $T = [-0.011 \quad 0 \quad 0.011]^T$ meters and there was no rotation.

Interpolating the image intensity values along the epipolar lines, it is possible to rectify the two images, thus obtaining images I_1^R and I_2^R, where the epipolar lines are collinear and parallel to the horizontal axis

$$\forall x, y \quad I_2^R(x, y) = I_1^R(x + \frac{T'}{Z}, y) \tag{4.10}$$

where the new translation vector is $T' = \sqrt{T_x^2 + T_z^2}$ and the new plane parameters are $(\alpha', 0, \gamma')^T = R_{RECT}(\alpha, 0, \gamma)^T$ with R_{RECT} being the rectification (rotation) matrix.

Combining Eqs. 4.9 and 4.10 and dropping for simplicity the prime notation we obtain

$$\forall x, y \quad I_2^R(x, y) = I_1^R((1 + \alpha T)x + \gamma T, y), \tag{4.11}$$

Table 4.1: Phase Correlation Concept

- Let 2D signals s_1 and s_2 be related by a translation (x_0, y_0) only, i.e.

$$s_2(x, y) = s_1(x - x_0, y - y_0)$$

- Their corresponding Fourier transforms are related by a phase shift which encodes the translation, i.e.

$$\mathcal{S}_2(u, v) = e^{-2\pi i(ux_0 + vy_0)}\mathcal{S}_1(u, v)$$

- The phase shift can be extracted from the Normalized Cross-power Spectrum of the two signals, which is defined as

$$NCS = \frac{\mathcal{S}_1(u, v)\mathcal{S}_2^*(u, v)}{|\mathcal{S}_1(u, v)\mathcal{S}_2^*(u, v)|} = e^{2\pi i(ux_0 + vy_0)}$$

- Thus, the inverse Fourier transform of NCS is a delta function around the translation point $(-x_0, -y_0)$

$$\mathcal{F}^{-1}\{NCS\}(x, y) = \delta(x + x_0, y + y_0)$$

We can estimate α and γ using phase correlation (Table 4.1) between the signals along the set of two epipolar lines in two steps [52]. First, we estimate α using phase correlation on the magnitude of the Fourier transform of the two signals in logarithmic coordinates (Eq. 4.16). Then, we warp the signals, using the estimate

for α, so that only the translation component is present. Finally, we estimate γ using phase correlation on the warped signals (Eq. 4.18). The complete algorithm along with the equations are presented in Alg. 4.2.

While the algorithm presented here, solves for two (α, γ) of the three plane parameters, it is possible to obtain all three parameters by performing a geometric transformation on the variables and exploiting 2D phase correlation.

4.4.2 Implementation details

In Figs. 4.6 and 4.7 we present the results of applying this method to a series of images obtained by the left side camera of our robot. In this experiment, we used 81 epipolar lines. The red crosses denote the distance and slant estimates for each pair of frames. While slant estimation is quite accurate, still the line method provided superior results. On the other hand, this method outperformed both the line based technique and the normal flow based technique (described in Section 4.5) in the distance estimation.

Another advantage of the method is its computational simplicity. Thus, the unoptimized Matlab code runs in ~ 1.5 seconds for an image of 81×1024 pixels (i.e., 81 epipolar lines of 1024 pixels each), with most of the time spent on warping the 2 signals in order to compute Eq. 4.17.

Algorithm 4.2 Estimate Plane Parameters α, γ

Input:

I_1^R, I_2^R : Image signals along Epipolar Lines

T :Translation

Output:

α, γ : Plane parameters

Algorithm:

- Signals along the epipolar line y

$$\forall x, \quad I_2^R(x,y) = I_1^R((1+\alpha T)x + \gamma T, y) \tag{4.12}$$

- Compute the Fourier Transform $(\mathcal{I}_1^R, \mathcal{I}_2^R)$ of I_1^R, I_2^R

$$\mathcal{F}_{x,y}\{I_2^R\}(u,v) = \frac{e^{2\pi i \frac{\gamma T}{1+\alpha T}u}\mathcal{F}_{x,y}\{I_1^R\}(\frac{u}{1+\alpha T}, v)}{|1+\alpha T|} \tag{4.13}$$

- Consider the Magnitude of $\mathcal{I}_1^R, \mathcal{I}_2^R$ and logarithmically transform (u, v)

$$|\mathcal{I}_2^R(\log u, v)| = \frac{|\mathcal{I}_1^R(\log u - \log(1+\alpha T), v)|}{|1+\alpha T|} \tag{4.14}$$

- Compute the Normalized Cross-power Spectrum (NCS_1) of $|\mathcal{I}_1^R|$, $|\mathcal{I}_2^R|$

$$NCS_1(\eta, w) = e^{2\pi i \eta \log(1+\alpha T)} \tag{4.15}$$

- Compute α taking the Inverse Fourier transform of NCS_1

$$\alpha = \frac{e^{u - argmax(\mathcal{F}^{-1}\{NCS_1\})} - 1}{T} \tag{4.16}$$

- Take the Normalized Cross-power Spectrum NCS_2 of $\mathcal{I}_1^R(\frac{u}{1+\alpha T}, v), \mathcal{I}_2^R(u,v)$ from Eq. 4.13

$$NCS_2(u,v) = e^{-2\pi i \frac{\gamma T}{1+\alpha T}u} \tag{4.17}$$

- Compute γ

$$\gamma = -\frac{(1+\alpha T) argmax(\mathcal{F}^{-1}\{NCS_2\})}{T} \tag{4.18}$$

4.5 Plane parameters from normal flow

4.5.1 Theory

As described before, $\overrightarrow{N} = (\alpha, \beta, \gamma)^T$ denotes a plane in the 3D world and $\overrightarrow{P} = (X, Y, Z)^T$ a point on that plane ($\overrightarrow{P} \cdot \overrightarrow{N} = 1$) and Eq. 4.1 is valid. When the camera moves with instantaneous rotational velocity $\overrightarrow{\Omega} = (\Omega_x, \Omega_y, \Omega_z)^T$ and translational velocity $\overrightarrow{t} = (t_x, t_y, t_z)^T$ the relative motion of the point is $V(\overrightarrow{P}) = -\overrightarrow{t} - \overrightarrow{\Omega} \times \overrightarrow{P}$. The corresponding motion of the image point $\overrightarrow{p}$ is

$$\begin{pmatrix} \frac{dx}{dt} \\ \\ \frac{dy}{dt} \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} t_z x - t_x f \\ \\ t_z y - t_y f \end{pmatrix} + \tag{4.19}$$

$$\begin{pmatrix} \Omega_z y - \Omega_y f + \frac{\Omega_x xy - \Omega_y x^2}{f} \\ \\ -\Omega_z x + \Omega_x f + \frac{-\Omega_y xy + \Omega_x y^2}{f} \end{pmatrix}. \tag{4.20}$$

Substituting equations (4.1) and (4.20) into the image brightness consistency constraint

$$\frac{\partial I}{\partial x} \cdot \frac{dx}{dt} + \frac{\partial I}{\partial y} \cdot \frac{dy}{dt} + \frac{\partial I}{\partial t} = 0, \tag{4.21}$$

we obtain an equation bilinear in the motion parameters and the plane parameters. Note that $I(x, y, t)$ represents the image intensity at point (x, y) and time t. In our case we have restricted motion (i.e. $\Omega_x = \Omega_z = 0$ and $t_y = 0$), so we can further

simplify the equation

$$
\begin{aligned}
& A(x\ y\ f)(\alpha\ \beta\ \gamma)^T = B \text{ ,where} \\
& A = \tfrac{I_x}{f}(xt_z - ft_x) + \tfrac{I_y}{f} yt_z, \\
& B = I_x f\Omega_y + \tfrac{\Omega_y}{f}(I_x x^2 + I_y xy) - I_t
\end{aligned}
\qquad (4.22)
$$

According to Eq. 4.22, knowing the motion parameters, the camera intrinsic parameters (i.e., focal length and principal point) and the image intensity derivatives, plane estimation amounts to solving a linear system of equations for the parameters (α, β, γ).

4.5.2 Implementation

To calculate the normal flow we used the gradient based method of Lucas and Kanade ([68]) using the filtering and differentiation kernels proposed by Simoncelli ([69]) on 5 consecutive frames. For performance reasons, we first reduced the size of the image by one quarter, so we are computing the gradients on a 256×192 array (as opposed to the whole 1024×768 original images). The image size reduction has the additional advantage of reducing the pixel displacement between successive frames, thus resulting in more accurate results for plane estimation. The unoptimized Matlab version of the code runs in ~ 0.4 seconds on our testbed, with most of the time spent in computing the spatial and temporal gradients.

In Figs. 4.6 and 4.7 we also display the results of running the normal flow based plane estimation algorithm in the same test sequence used for the previous

methods. It is clear that this method is less accurate in distance and slant estimation compared to the texture and the line method, respectively for that specific image sequence. This is due to the lack of good image features to track in our environment.

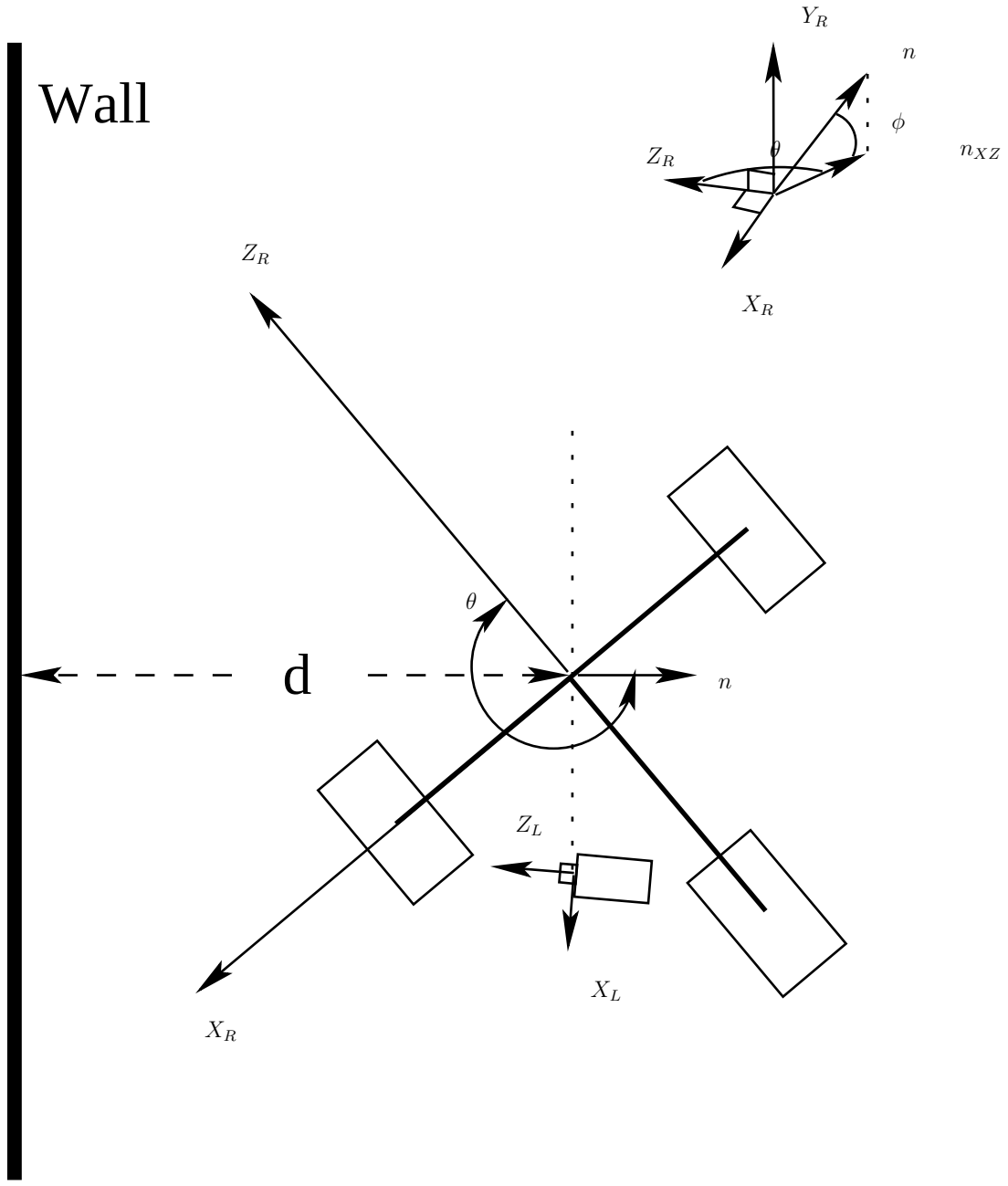
4.6 Extended Kalman Filter

Integration of the individual measurements over time is performed using an extended Kalman filter (EKF). First, let us define a robot-centric coordinate system $O_R X_R Y_R Z_R$ as follows (Fig. 4.5); the center O_R coincides with the midpoint of the two front wheels of the robot, the X_R axis points to the left wheel of the robot, the Y_R axis points upwards and the Z_R axis forward.

As state variables for the Kalman filter we use the *distance/slant/tilt* parametrization of the plane, $\mathbf{S}(t) = [d, \theta, \phi]^T$. If we denote $\overrightarrow{n}_{XZ}$ the projection of $\overrightarrow{n}$ on the $Y = 0$ plane, then we define the *slant* θ to be the angle between the Z_R axis and $\overrightarrow{n}_{XZ}$, as shown in Fig. 4.5. *Tilt* ϕ is the angle between the Y component of n and the XZ plane. Thus the transformation between the two different parametrization is

$$\begin{bmatrix} d & = & \frac{1}{\sqrt{\alpha^2+\beta^2+\gamma^2}} \\ \theta & = & \arctan\left(\frac{\alpha}{\gamma}\right) \\ \phi & = & \arccos\left(\frac{\beta}{d}\right) \end{bmatrix}$$

Assuming that the control vector $\mathbf{U}(t)$ consists of the instantaneous translational and rotational velocities of the robot $(v(t), \omega(t))$ respectively and Δt denotes a time interval, the evolution of the system over time can be described as



(a) Robot Sketch

Figure 4.5: The distance and angle θ between the robot and the wall are defined with respect to a coordinate system attached to the robot. The surface normal projected on the $X - Z$ plane (n_{XZ}) is also displayed.

$$\mathbf{S}(t+\Delta t) = \mathbf{F}(\mathbf{S}(t), \mathbf{U}(t)) \Leftrightarrow$$

$$\left[\begin{array}{lcl} d(t+\Delta t) & = & d(t) + v(t)\cos\theta(t)\Delta t + \epsilon_{11} \\ \theta(t+\Delta t) & = & \theta(t) - \omega(t)\Delta t + \epsilon_{12} \\ \phi(t+\Delta t) & = & \phi(t) + \epsilon_{13} \end{array}\right], \tag{4.23}$$

where we use the assumption that $\cos\theta(t) \simeq \cos\theta(t+\Delta t)$, i.e. the rotational velocity $\omega(t)$ is small and approximately constant over Δt and the discretization step Δt is also small. Furthermore, we denote with ϵ_{1i} the errors in the state prediction (with covariance $\mathbf{Q}$).

Our measurement vectors (Z_1, Z_2, Z_3) consist of the plane parameters calculated using the different methods described in Sections 4.3, 4.4 and 4.5 respectively, converted to the distance/slant/tilt parametrization. We consider the combined measurement to be a weighted linear combination of the individual measurements i.e., $Z(t) = \sum_{i=1}^{3} C_i Z_i$, where the weights C_i encode the (inverse) uncertainty of the estimates using different methods, which we derived as follows.

The line module bases the accuracy of the plane estimation on how well it detects and localizes the line. The harmonic texture module is using the magnitude of the Inverse Fourier transform of the Normalized Cross-power Spectrum (Eqs. 4.16, 4.18) and the normal flow module is using the condition number of the linear system (Eq. 4.22).

The system evolution (Eq. 4.23) is not linear with respect to the state vector $\mathbf{S}(t)$ and the control vector $\mathbf{U}(t)$. That's why we need to use an extended Kalman

filter and linearize the equations by considering the Jacobian matrix as shown in Table 4.2.

Table 4.2: extended Kalman Filter Equations

Jacobian of system evolution with respect to the state vector $S(t)$

$$\mathbf{A}(t) = \begin{bmatrix} 1 & -v(t)\sin\theta(t)\Delta t & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.24}$$

Jacobian of system evolution with respect to the control vector $U(t)$

$$\mathbf{W}(t) = \begin{bmatrix} \cos\theta(t)\Delta t & 0 & v(t)\cos\theta(t) \\ 0 & -\Delta t & -\omega(t) \\ 0 & 0 & 0 \end{bmatrix} \tag{4.25}$$

State prediction equations (Mean $\hat{\mathbf{S}}$ and Covariance $\hat{\mathbf{P}}$)

$$\begin{bmatrix} \hat{d}(t+\Delta t) & = & \bar{d}(t) + \bar{v}(t)\cos\bar{\theta}(t)\bar{\Delta}t \\ \hat{\theta}(t+\Delta t) & = & \bar{\theta}(t) - \bar{\omega}(t)\bar{\Delta}t \\ \hat{\phi}(t+\Delta t) & = & \bar{\phi}(t) \end{bmatrix}. \tag{4.26}$$

$$\hat{\mathbf{P}}(t+\Delta t) = \mathbf{A}(t)\bar{\mathbf{P}}(t)\mathbf{A}(t)^T + \mathbf{W}(t)\mathbf{Q}(t)\mathbf{W}(t)^T \tag{4.27}$$

Kalman Gain $\mathbf{K}$

$$\mathbf{K}_i(t) = \hat{\mathbf{P}}(t)(\hat{\mathbf{P}}(t) + \mathbf{R}(t))^{-1} \tag{4.28}$$

Measurement update equations (Mean $\bar{\mathbf{S}}$ and Covariance $\bar{\mathbf{P}}$)

$$\bar{\mathbf{S}}(t+\Delta t) = \hat{\mathbf{S}}(t+\Delta t) + \mathbf{K}(t)(\mathbf{Z}(t+\Delta t) - \hat{\mathbf{S}}(t+\Delta t)) \tag{4.29}$$

$$\bar{\mathbf{P}}(t+\Delta t) = (\mathbf{I} - \mathbf{K}(t))\hat{\mathbf{P}}(t+\Delta t) \tag{4.30}$$

4.6.1 Results

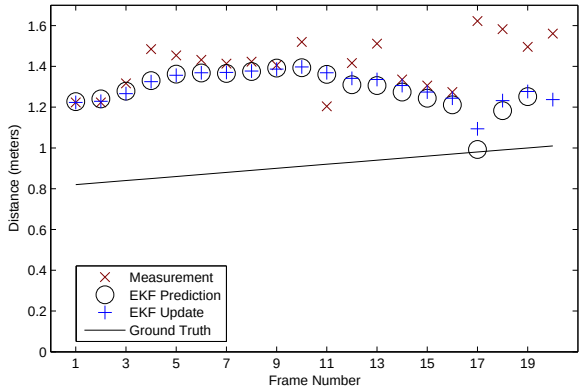
Figs. 4.6 and 4.7 depict the results when we combined the line, texture and normal flow methods, respectively with the odometry measurements using the EKF. More specifically, in these figures, black circles denote the prediction about the current

state using only the previous state and dead reckoning information (Eq. 4.26), while blue pluses denote the final prediction of the state after the measurements from each individual module are also considered (Eq. 4.29). It is clear that integration of measurements over time *significantly* improves the accuracy and robustness of the method.

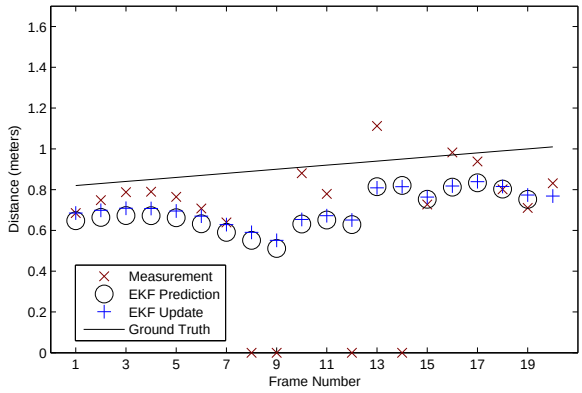
4.7 Motion Control

An important part of any navigation system is the motion control subsystem. In this particular setting the goal is to move along the corridor avoiding the obstacles that might lie ahead of us. The motion control strategy described below refers to the “wall-following" behavior. Using the same policy one could implement the “centering" behavior.

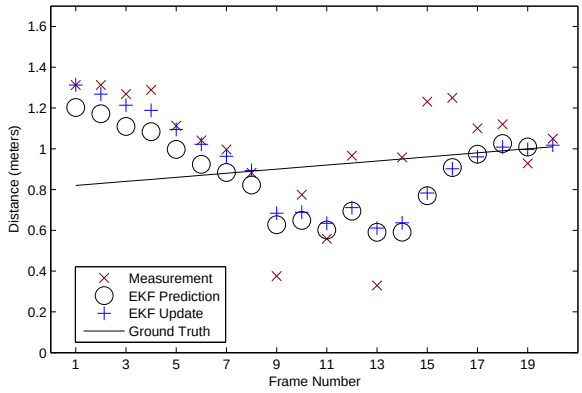
Let's define the input to the motion control algorithm to be the state vector of the Kalman filter, that denotes the position of the left wall with respect to the robot. Ideally, we want the robot to remain at a constant distance (denoted with D_C) from the wall, thus following the line $\mathcal{L}_C$ as shown in Fig. 4.8. In practice, the robot's trajectory is restricted by motion dynamics as well as the constraint that the rotational and translational velocities should remain constant, while the camera is recording the frames. As a consequence, the system is only allowed to perform small motion changes between two successive frames, thus it is hard to follow the virtual line. Instead, a point $\mathcal{P}$ along the line $\mathcal{L}_C$ is picked and the robot's motion is regulated accordingly, so that it approaches $\mathcal{P}$. Next we describe how to do this.



(a) Line module

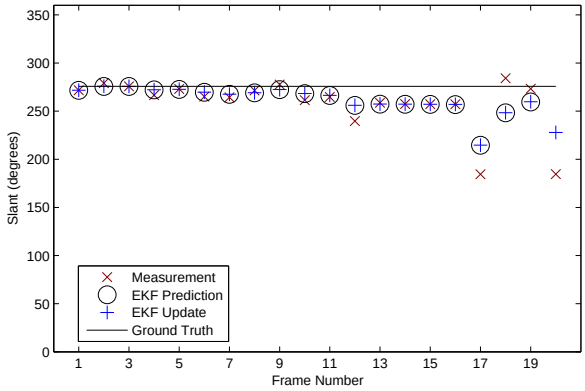


(b) Texture module

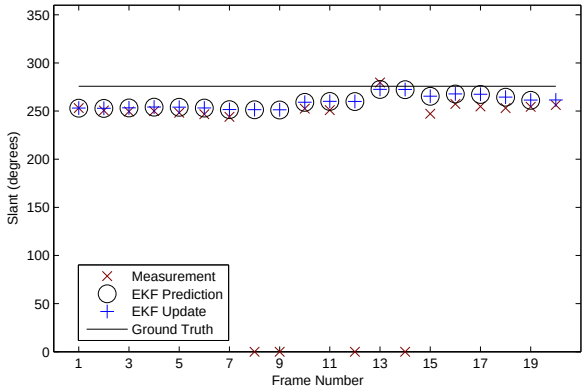


(c) Normal flow module

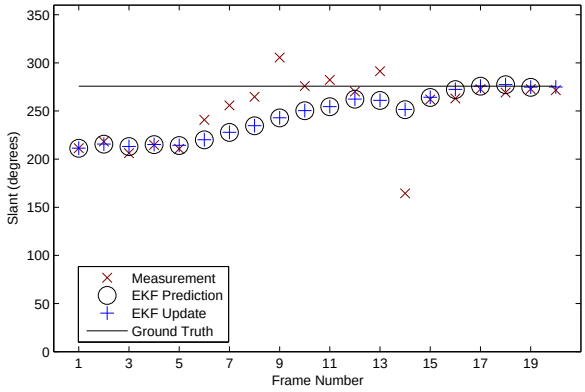
Figure 4.6: The distance results of one test run. We display the estimates of each module with a cross, the extended Kalman filter prediction (Eq.) with a circle and the final estimate after integration with the measurement (Eq.) with a plus sign. In some frames no reliable estimate could be obtained using the harmonic texture method(second column). In these cases, we display the red cross on the bottom of the corresponding figure. Also note the first EKF update is based solely on image estimates.



(a) Line module



(b) Texture module



(c) Normal flow module

Figure 4.7: The slant results of one test run. We display the estimates of each module with a cross, the extended Kalman filter prediction (Eq.) with a circle and the final estimate after integration with the measurement (Eq.) with a plus sign. In some frames no reliable estimate could be obtained using the harmonic texture method(second column). In these cases, we display the red cross on the bottom of the corresponding figure. Also note the first EKF update is based solely on image estimates.

Let's assume that point $\mathcal{P}$ is y_P meters away from the robot along the line $\mathcal{L}_C$ and forms an angle ψ as shown in Fig. 4.8. Furthermore, the robot is situated x_P units away from $\mathcal{L}_C$ and is moving with instantaneous translational and rotational speed $v(t), \omega(t)$ respectively. Note that the translational velocity is always along the direction of the Z-axis of the robot and the rotational velocity is around the Y-axis. Then, we have:

$$\psi = \arctan(\frac{y_P}{x_P}) \tag{4.31}$$

$$\xi = \theta - \pi - \psi \tag{4.32}$$

The line segment L_{RP} has length $D = \sqrt{x_P^2 + y_P^2}$. An approximation of the time

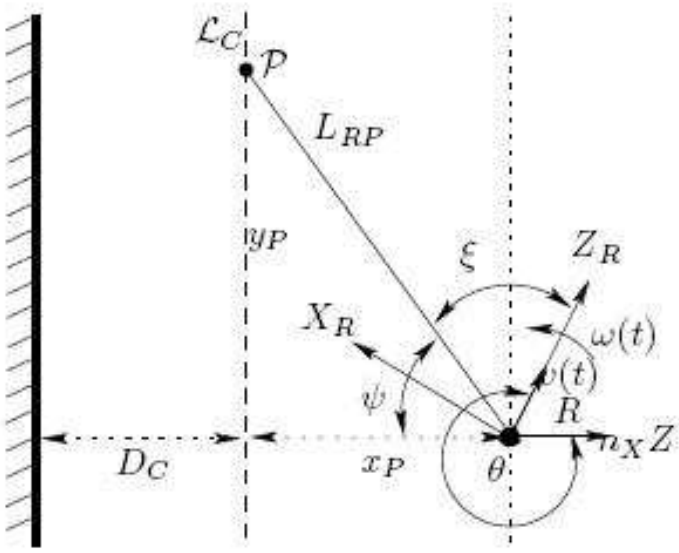


Figure 4.8: The robot R is moving with translational and rotational velocities $v(t)$, $\omega(t)$ respectively, while it is located x_P units away from the virtual line $\mathcal{L}_C$.

that is required by the robot to reach point $\mathcal{P}$ is $\Delta t = \frac{D}{v(t)}$. The new rotational velocity ($\omega(t + \Delta t)$) of the robot should be:

$$\omega(t + \Delta t) = \frac{\xi}{\Delta t} = v(t)\frac{\theta - \pi - \arctan\frac{y_P}{x_P}}{\sqrt{x_P^2 + y_P^2}} \tag{4.33}$$

4.8 Experiments

We have used the robotic platform ER1 from Evolution Robotics. On top of it, we have placed a front and two side Firewire cameras (SONY XCD-X700). The side cameras form angles ($\sim 45^o, \sim -45^o$) with the front camera as shown in Fig. 4.9. In the following experiments we used the left side camera and the front camera. We run the texture based as well as the normal flow based code on the left side camera and the line-based code on the front camera.

The goal of the experiments is to convey two messages;

- The accuracy and robustness of the system significantly increases with the *integration* of individual measurements from *different subsystems* over *time*.
- When using all the methods the robot is able to move along a mostly textureless corridor.

4.8.1 Constant Distance Experiment

The goal of this first experiment was for the robot to move a distance of 20 meters along a corridor without hitting the side walls. The corridor had a width of 1.8 meters, so we instructed the robot to try to maintain a distance of 0.9 meters from the

(a) Photo of robot

Figure 4.9: The ER1 robot equipped with 3 Firewire cameras. The height of the robot is $\sim$ 70 cm. In the background, part of the corridor, where we conducted some experiments, is shown. All the walls and doors are textureless and there exist significant specular highlights on both the walls and the floor caused by the light sources.

left, while moving with velocity 5 cm/sec. The initial orientation of the robot with respect to the wall varied from 0^o (parallel to the wall) to -20^o (moving away from the left wall) and $+20^o$ (moving towards the wall). We made multiple runs each time activating a different submodule with and without integrating the measurements with dead reckoning using the EKF. Finally, we performed the experiment using all the submodules together. The results are presented in Fig. 4.10. It is clear that each individual module in isolation performs poorly (with the exception of the line module). Integrating the measurements of a single module over time (using the EKF) greatly improves the robustness of the method. Finally, combining the measurements from different submodules, provides the most robust setting.

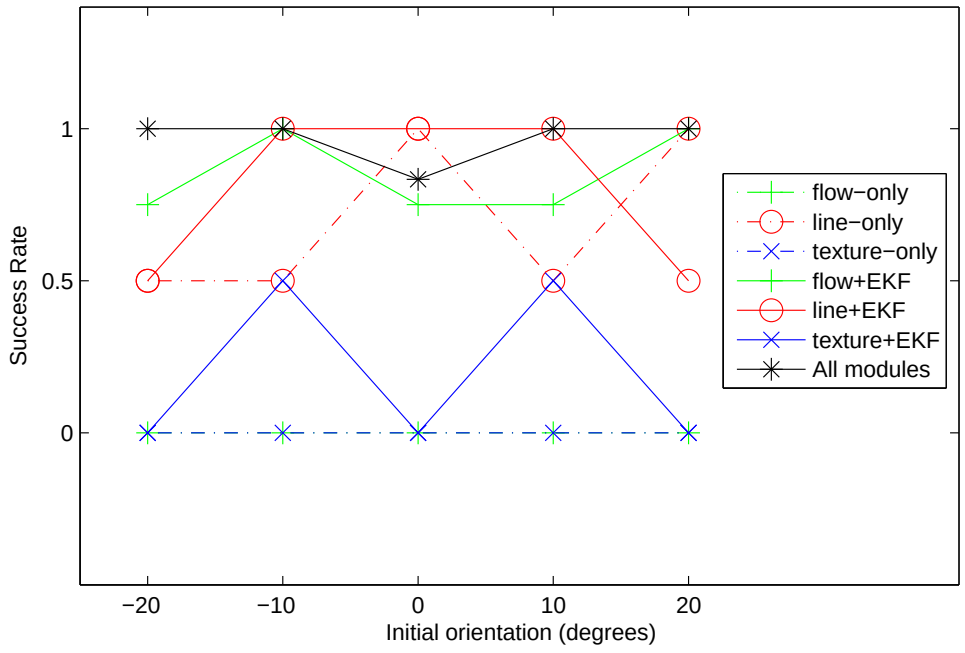


Figure 4.10: Percentage of times that the robot was able to move than 20 meters without hitting the side walls.

4.8.2 Average Distance Experiment

In this experiment we let the robot move on the corridor (still trying to maintain a distance of 0.9 meters from the left wall) with velocity 5 cm/sec, and measured the average distance traversed before the hitting the wall. We performed the experiment multiple times activating a different module or combinations of modules. The results, namely the average distance for each combination, are presented in Fig. 4.11. Again, we observed that a single module performs very poorly (with the exception of the line module), while combining modules together and integrating the estimates over time greatly improves the result. When the average distance is larger than 20 meters, it indicates that the robot is approaching the end of the corridor and thus we had to terminate the specific run.

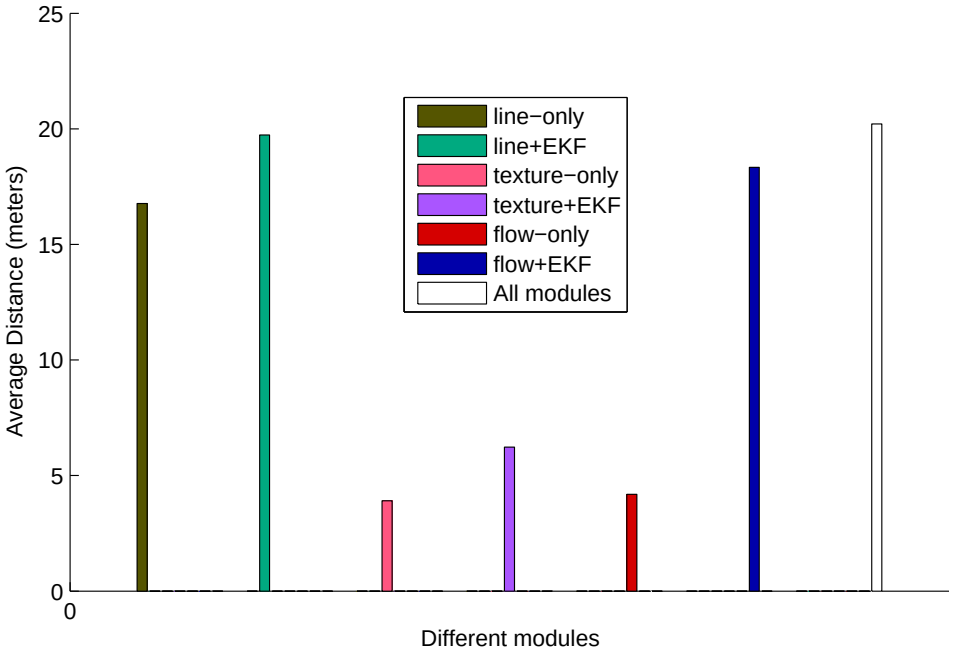


Figure 4.11: Average distance that the robot was able to move using measurements from a single or multiple modules.

4.9 Conclusions

In this chapter we presented two new methods for computing the 3D structure of a piece-wise planar scene from video. We also used an existing method for 3D shape estimation based on normal flow. The three methods base their estimation on complementary information. More specifically, while the normal flow technique considers individual features (i.e. sharp intensity changes) within the object, the texture method considers the whole area within it. The line method, on the other hand, uses the boundaries of an object. Depending on the case, we expect at least one of the methods to provide accurate measurements. For example, when we observe a mostly uniformed colored object, we anticipate that the line method will be able to accurately track the boundary of it and produce accurate results, while the remaining two modules will fail. On the other hand, when the object is highly textured, the line method might not be able to locate the boundaries accurately, but the two other methods will produce good results. For that reason, we emphasize that the integration of all three modules is the right approach, if one wants to build a robust system. For similar reasons, integration of the individual measurements over time is equally important. In this paper, we use odometry measurements from the wheel encoders, but we might as well estimate the motion from the video (*visual odometry*, also known as *ego-motion* estimation [70],[71],[72]) or using other sensors. We present experiments in the context of visual navigation on indoor environments and verify that the combined usage of all three modules produces a more robust system.

This chapter is complementary to the previous chapter on color based segmentation in a sense that one basic assumption for estimating the structure is that we have solved the segmentation problem and thus we know the boundaries of the planar surfaces. In order for the robot to navigate in more complex environments, we need to incorporate a scene segmentation scheme into this framework. In later chapters we argue how this integration can be performed.

Chapter 5

Towards Surface Segmentation

5.1 Introduction

In the previous chapters we focused on color based image segmentation (chapters 2 and 3) and on surface normal estimation (chapter 4). Both problems constitute two important components of a system that performs segmentation into surfaces. In this chapter we conclude our thesis by discussing how these two components can be combined[1]. We also touch on the topic of actively controlling the image acquisition process to facilitate the segmentation.

We use the term surface segmentation (or segmentation into surfaces) to denote the geometry inspired segmentation where adjacent pixels with similar surface vectors are grouped together. The term surface vector is used to denote both the surface orientation (i.e. surface normal) and its distance from the focal point. In that definition region boundaries are identified as discontinuities in surface vectors, caused either by a discontinuity in the distance (i.e. occlusion) or by a discontinuity in the orientation.

We have chosen the above definition because it turns segmentation into a well defined problem. Generally, most definitions of image segmentation are object

[1]Since we are interested in segmentation into surfaces (and not on visual guided navigation as in chapter 4) we choose to estimate the plane induced homographies of sets of points in two views. This is arguably a relatively "easier" problem than the full 3D reconstruction of a scene.

Figure 5.1: An image of an office chair. Notice that there is a smooth normal transition from the pixels belonging to the back of the chair to the pixels belonging to the bottom of it.

oriented and thus ambiguous and ill-defined. The ambiguity is partially due to the fact that multiple meaningful segmentations exist for the same image at different levels of details. A person, for example, can consider a laptop computer as a single object, or further segment it into the LCD display and the keyboard. Further subdivision of the keyboard to its keys is also valid. That is one of the reasons for researchers to suggest that the proper segmentation is task and domain specific [3]. One way to deal with the ambiguity is to accept multiple segmentations as valid. This is the path chosen by Martin et al. [36] and Alpert et al. [8]. When building their image segmentation database they included multiple possible segmentations of the same image, each one produced by a different person. The major cause of the ambiguity though is that the concept of an "object" is by itself ill-defined and subjective. Hence, image regions corresponding to objects are by default subjective as well. The use of a well defined geometric feature, such as the surface vector, makes the segmentation a better defined problem.

Unfortunately, the issues associated with different segmentations at different

image resolutions are not eliminated even with that definition. It is absolutely natural and often happens in practice that the computation of the surface normals at different image resolutions might lead to different segmentations. Borrowing an example from the introductory chapter, the surface normal based segmentation of the office chair of Fig. 5.1 might lead to two separate segments, one for the back and one for the bottom of the chair, or to one segment containing both the back and the bottom of the chair, depending on how coarse or fine is the computation of the surface vectors.

Furthermore, the computation of the surface vectors itself is hard and not very accurate. One needs to hypothesize a model for the surface of the surrounding area of a pixel in order to measure its surface vector. This fact leads to a chicken and egg problem because in order to segment based on surface vectors one needs to assume that the surface vectors in the surrounding area are similar i.e., to assume that the surrounding area belongs to the same segment. Apparently this model breaks in areas near surface normal discontinuities. In the following sections we assume that *the* $3D$ *world consists of planar patches* and hence, the surface normals within a patch are constant. As we will show below, even in this case the accurate computation of the surface normals is not trivial and in most cases not even possible.

The keyword "*active*" in this context refers to the idea that the camera motion can be controlled (up to a certain degree) by the process that performs the segmentation. We motivate the discussion of this problem by the setting of Fig. 5.2 where the cameras position and orientation can be changed in two ways; a) by moving the whole mobile platform and b) by moving the Pan and Tilt unit. The

Figure 5.2: Our mobile robot. We used the Erratic mobile platform as the basis and installed on top a laser and sonar range sensors, a Pan and Tilt unit and a quad stereo system.

motion of the whole robot leads to a large translational and rotational motion of the cameras (with high uncertainty as reported by the wheel encoders). The motion of the PTU, on the other hand, causes small translational and rotational motion to the cameras that can be estimated with high accuracy. Overall, the system allows for an almost unrestricted control of the cameras position and orientation. In this case an interesting question is how specific motions and poses of the camera can simplify the estimation of surface vectors and subsequently surface segmentation.

In the following section we summarize related work on structure from motion and active vision. Then, we present the related theory on homography estimation from two images along with one lemma that can be used to predict the quality of this estimation. Sec. 5.4 touches on the problem of merging image regions with similar homographies and splitting regions with many outliers. We argue that the color based segmentation framework that we presented on chapters 2 and 3 can be used to guide the merging and splitting process. Finally, we briefly illustrate our

idea on how an active camera control system can be constructed.

5.2 Related Work

The structure from motion problem is a prominent research area in computer vision and as such has been studied extensively. In the most common formulation the goal is to compute the structure of the scene (i.e., the distance of every pixel from the focal point) and the motion of the camera (or the motion of the scene objects if they are moving independently). Since the early 80's where the existence of the solution has been established [73, 74, 70, 75], a great number of researchers have tried to devise algorithms that work well under realistic situations were noise is present. Any computer vision textbook, such as [76], contains a description of the basic algorithms along with the related bibliography. In the recent years bundle adjustment i.e., a collection of optimization methods from the photogrammetry and geodesic literature tailored to solve the structure from motion problem, was imported in computer vision [77]. These methods that are used to refine an initial estimate for the structure of the scene and the motion of the camera are shown to produce real-time, high quality scene reconstructions [78][2]. On a parallel track a number of theoretical studies on the structure from motion problem have been conducted [79, 80, 81, 82, 83, 84, 85]. Their goal is to understand and describe the inherent ambiguity in recovering structure and motion, discover the configurations

[2]Since bundle adjustment techniques optimize over a non convex domain they suffer from the same convergence problems as all other non-convex methods. Thus, the initial estimate feeded to the system should be close to the global minimum, otherwise the methods will converge to a local minimum different from the true solution.

that makes this recovery ill-conditioned and perform error analyses for different types of noisy input.

In the following sections we concentrate on a slightly different formulation of the problem. Instead of solving for the structure and motion of individual pixels we assume the image to be piecewise planar and focus on the recovery of the homographies induced by the planar patches in two views. Methods for computing homographies have been extensively covered by Hartley and Zisserman in their book [43]. Since then, a number of studies have been performed on how to identify planar patches on a scene from the homography computation of individual image feature points and merge them together [86, 87, 88, 89, 90, 91, 92, 93]. With respect to previous work our proposed approach presented below also takes into account the results of color based segmentation at different levels of detail in order to group feature points together. The rational is similar to the one used by state of the art stereo algorithms [94, 1, 2]; namely color based segmentation is an additional cue that can be used to guide the grouping of feature points before computing the homographies. At this point we should mention that a few other problems require the computation of 2D homographies (for a presentation of these problems refer to [43]) and our covariance estimation theory (i.e., Lemma 5.2) is actually similar to the method suggested in [95] for measuring the 3D properties of objects from 2D images.

All the previous approaches follow the "vision as a recovering process" paradigm of Marr [96]. In the late 80's a different paradigm under the name of Active [97, 98], Animate [99] and Purposive vision [100] has been introduced. Under this doctrine,

image understanding and computer vision should also study the process of selective acquisition of data in space and time. More specifically, depending on the goal of the visual system a proper strategy for controlling the image acquisition process can significantly improve the results of the visual computation or even make ill-defined problems (e.g. structure from motion) well defined. Since its conception a series of studies following this paradigm have been published, some of them discussing the visual capabilities that an "intelligent" system should have [101, 102, 103, 104], while others focusing on the optimal camera motion strategy for specific tasks e.g. [105, 106]. Still, the amount of work following this model is relatively small. Of course there are theoretical problems related to designing the proper visual tasks and camera control strategies, but we believe that the main issues holding back this paradigm had been of a practical nature thus far. The image acquisition hardware (mobile platforms, cameras with pan and tilt capabilities, mechanical arms etc) was either too expensive, too sensitive or too bulky to allow the construction of a real active visual system. Most importantly the computers were not fast enough to allow real time image processing. In the recent years with the introduction of multicore CPUs and GPUs, this situation has been reversed, so we expect this paradigm to gain momentum once more in the next years.

5.3 Homography estimation of planar surfaces

Let us assume that we can detect a set of n image points belonging to a single world plane and track them over two frames. We denote the homogeneous, image

coordinates of a point on the first and second frame with $\mathbf{x}_i = (x_i,\ y_i,\ 1)^T$, $\mathbf{x}'_i = (x'_i,\ y'_i,\ 1)^T$ respectively. In the remaining chapter we will use the bold notation for vectors only. All the points belong to the same plane, thus there exist a 3×3 matrix H, know as homography matrix, that corresponds the coordinates of the point in the first and second plane, namely $\mathbf{x}' = H\mathbf{x}$. The $' =$ sign here does not denote equality. The vectors $\mathbf{x}'$ and $H\mathbf{x}$ have the same direction, but may differ in magnitude by a non-zero scale factor. This can be expressed in terms of the vector cross product as $\mathbf{x}' \times H\mathbf{x} = \mathbf{0}$. The homography matrix is unique up to a scale and thus has 8 degrees of freedom. Each point contributes two equations thus at least 4 points are needed to compute H. If $n > 4$, the matrix is overdetermined and H is computed by a suitable minimization scheme. Two are the dominant estimation methods, the *homogeneous solution* of *minimizing the algebraic distance* and the *non-linear solution* of *minimizing the geometric distance*.

The first method uses the SVD decomposition to solve a homogeneous linear system. It has the advantage over the second method of being fast and convex (thus the minimization finds a global minimum). On the other hand, the objective function is not geometrically meaningful, and thus the result might be bad.

The second method uses an objective function that computes the sum of the Euclidean distances between the measured and mapped points. This quantity is meaningful and corresponds to the measurement error. On the other hand, the minimization is not convex and there is no closed form solution. Thus, iterative methods should be employed to solve the system. Depending on the initialization the method can be slow and converge to a local minimum. In practice, the first

method is used first to compute a good initial estimate of H that is further refined using the second method.

In the following analysis we use the first method because it allows us to compute the covariance of the estimated homography H. Here is a brief description of the method. For more details about both methods the interested reader should consult Hartley and Zisserman's book [43].

First, we write the homography matrix H in vector form as

$$\mathbf{h} = (h_{11},\ h_{12},\ h_{13},\ h_{21},\ h_{22},\ h_{23},\ h_{31},\ h_{32},\ h_{33})^T.$$

With the proper algebraic manipulation of the homography equation $\mathbf{x}' \times H\mathbf{x} = \mathbf{0}$ we get a linear homogeneous equation for the computation of the homography vector

$$A\mathbf{h} = 0, \tag{5.1}$$

where

$$A = \begin{bmatrix} 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ -y'_1x_1 & -y'_1y_1 & -y'_1 & x'_1x_1 & x'_1y_1 & x'_1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_n & -y_n & -1 & y'_nx_n & y'_ny_n & y'_n \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \\ -y'_nx_n & -y'_ny_n & -y'_n & x'_nx_n & x'_ny_n & x'_n & 0 & 0 & 0 \end{bmatrix}. \tag{5.2}$$

In the previous equation the third (and the sixth, the ninth etc) row is linearly dependent to the previous two rows so we can skip them and obtain a smaller equivalent matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_n & -y_n & -1 & y'_nx_n & y'_ny_n & y'_n \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \end{bmatrix}. \tag{5.3}$$

Eq. 5.1 as an optimization problem is expressed as

$$\arg\min_{\mathbf{h}} ||A\mathbf{h}||, \quad \text{s.t.} \tag{5.4}$$

$$||\mathbf{h}|| = 1 \tag{5.5}$$

The constraint $||\mathbf{h}|| = 1$ is necessary to avoid the obvious solution $\mathbf{h} = \mathbf{0}$. We use the following lemma to solve the problem.

Lemma 5.1. *The solution to a homogeneous minimization problem* $\arg\min_{\mathbf{X}} ||A\mathbf{h}||$ *subject to* $||\mathbf{h}|| = 1$ *is the eigenvector of the least eigenvalue of* A^TA.

Proof. Refer to [43], Appendix 3. □

The computation of the homography using the previous lemma is the first step. The second step is to compute an estimate on how accurate the computed homography is. Feature detection and localization, point mismatching, spatial quantization

and camera distortion errors directly affect the accuracy of the homography estimation. With the careful calibration of the camera the last error can be minimized. Robust point matching using RANSAC can solve the problem of point mismatching. Still the error in the precise detection and localization of the features cannot be avoided. These errors in the image coordinates of the points are usually modelled as random variables. Then the question is how these errors affect the computation of the homography vector $\mathbf{h}$.

The next lemma provides a way to compute the covariance of the estimated homography $\mathbf{h}$ with respect to the noise in the image coordinates of the detected features.

Lemma 5.2. *If we model the error in the localization of the feature points as independent Gaussian random variables with variance σ^2, σ'^2 for the features on the first and second frame respectively, the 9×9 covariance matrix of the homography is*

$$C_{\mathbf{h}} = JSJ^T \tag{5.6}$$

,where

$$J = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_9 \end{bmatrix} \begin{bmatrix} 0 & 0 & \dots & 0 & \mathbf{x}_1^T \\ 0 & \frac{1}{\lambda_1-\lambda_2} & \dots & 0 & \mathbf{x}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda_1-\lambda_9} & \mathbf{x}_9^T \end{bmatrix} \tag{5.7}$$

,with $\mathbf{x}_i$ the eigenvector corresponding to the i^{th} smaller eigenvalue λ_i of matrix

$A^T A$. *Matrix* S *is*

$$S = \sum_{i=1}^{n} (\mathbf{r}_{2i}^T \mathbf{r}_{2i} f_i^e + \mathbf{r}_{2i-1}^T \mathbf{r}_{2i-1} f_i^o + \mathbf{r}_{2i}^T \mathbf{r}_{2i-1} f_i^{eo} + \mathbf{r}_{2i-1}^T \mathbf{r}_{2i} f_i^{oe}) \tag{5.8}$$

with $\mathbf{r}_i$ *the* i^{th} *row of matrix* A *and*

$$\begin{aligned}
f_i^e = \;& \sigma^2[h_1^2 + h_2^2 - 2x_i'(h_1 h_7 + h_2 h_8)] + 2\sigma'^2(x_i h_7 h_9 + x_i y_i h_7 h_8 + y_i h_8 h_9) \\
& + (\sigma^2 x_i'^2 + x_i^2 \sigma'^2) h_7^2 + (\sigma^2 x_i'^2 + y_i^2 \sigma'^2) h_8^2 + \sigma'^2 h_9^2 \\
f_i^o = \;& \sigma^2[h_4^2 + h_5^2 - 2y_i'(h_4 h_7 + h_5 h_8)] + 2\sigma'^2(x_i h_7 h_9 + x_i y_i h_7 h_8 + y_i h_8 h_9) \\
& + (\sigma^2 y_i'^2 + x_i^2 \sigma'^2) h_7^2 + (\sigma^2 y_i'^2 + y_i^2 \sigma'^2) h_8^2 + \sigma'^2 h_9^2 \\
f_i^{oe} = f_i^{eo} = \;& \sigma^2[(h_1 - x_i' h_7)(h_4 - y_i' h_7) + (h_2 - x_i' h_8)(h_5 - y_i' h_8)].
\end{aligned}$$

Proof. A sketch of the proof is provided on appendix C. □

As expected the actual values of the homography matrix also affects the covariance matrix. Configurations that are almost ill defined as for example four points forming a straight line produce a large covariance matrix.

If we represent with $\bar{\mathbf{x}}_i' = (\bar{x}_i',\ \bar{y}_i',\ \bar{z}_i')^T$ the projection of point $\mathbf{x}_i$ on the second image using the homography H i.e., $\bar{\mathbf{x}}_i' = H\mathbf{x}_i$, then the *algebraic error* of the projection of the feature point of the first image into the second image is given by the formula

$$d_{alg}(\bar{\mathbf{x}}_i', \mathbf{x}_i') = \sqrt{(\bar{x}_i' - x_i')^2 + (\bar{y}_i' - y_i')^2 + (\bar{z}_i' - z_i')^2}. \tag{5.9}$$

The corresponding geometric error is given by the formula

$$d_{geom}(\bar{\mathbf{x}}'_i, \mathbf{x}'_i) = \sqrt{(\frac{\bar{x}'_i}{\bar{z}'_i} - \frac{x'_i}{z'_i})^2 + (\frac{\bar{y}'_i}{\bar{z}'_i} - \frac{y'_i}{z'_i})^2}. \tag{5.10}$$

After estimating the homography matrix it is possible to separate the points that belong to the plane based on their reprojection error.

Finally the following lemma relates the plane induced homography with the camera parameters and the surface normal.

Lemma 5.3. *Given the projection matrices for the two views of a camera with intrinsic parameters K*

$$P = K \cdot [I \mid \mathbf{0}] \qquad P' = K \cdot [R \mid \mathbf{T}] \tag{5.11}$$

where R, $\mathbf{T}$ represent the rotation and translation between the two views respectively and a plane defined by $\pi^T \cdot \mathbf{X} = 0$ with $\pi = (\nu^T,\ 1)^T$ (ν is the surface normal), then the homography induced by the plane is $\mathbf{x}' = H \cdot \mathbf{x}$ with

$$H = K \cdot (R - \mathbf{T} \cdot \nu^T) \cdot K^{-1}. \tag{5.12}$$

Proof. Appendix C. □

5.4 Merging and Splitting Image Segments

Two perspective views of a planar $3D$ surface are related by a homography. As we mentioned before there is a significant amount of work focusing on how to compute planar homographies and merge them together. The results of the state of the art algorithms are quite impressive, e.g. in [93] multiple planes belonging to different objects are detected on both indoor and outdoor image sequences. Still we think that one part of the plane identification process that has not received enough attention is how to select the groups of feature points used to compute the homography from. Over the years heuristics based on the proximity of the feature points and overall shape of the convex hull they form, have been used [93], but the usual approach is to try many quartets of feature points. As a result, the plane identification algorithms are usually quite slow (e.g. Amintabar and Boufama in [93] report a running time of 3.5 seconds for their optimized C code on 90 features). Here, we propose to use the results on color based segmentation to guide that process.

Fig. 5.3 synopsizes our approach. Starting from an image sequence we initiate two parallel computations. On one hand, we apply the KLT feature tracker [68, 107, 108] to detect and track a number of feature points. In order to eliminate spurious features we only take into account features that were tracked over multiple frames. On the other hand, color based segmentation using our MF+GAT algorithm (Sec. 3.4.3) at different granularities is performed. We combine the results of the two previous steps by grouping together features belonging to the same color segments. Using a robust estimation technique (RANSAC [109, 43]) we estimate

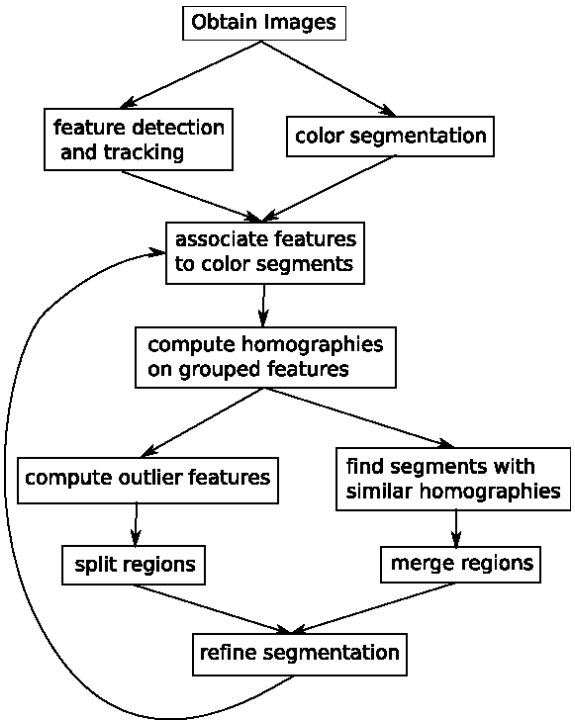


Figure 5.3: Our proposed scheme to address the 3D plane estimation problem on static image sequences by combining feature based homography estimation with color based segmentation.

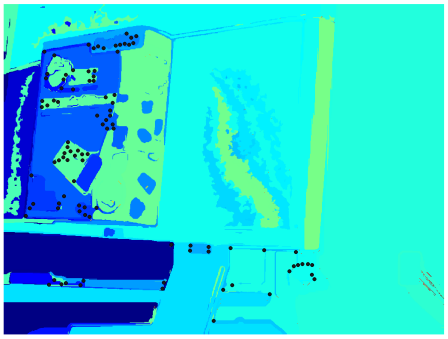
the homography associated with each color segment. Only the features that belong to the same image segment are used to compute its homography. Then, we run two parallel processes; a) for computing the outlier features and b) to find the segments with similar homographies. The former process is used to subdivide a color region, while the latter to merge regions together. Based on the results of both processes we can perform a more informed guess which is the correct local color resolution of our color based segmentation algorithm further refining the color segmentation with shape information.

In Fig. 5.4 we demonstrate the merging part of the algorithm. We obtained a sequence of 5 images using the cameras mounted on the robot. Then, we use a KLT tracker to detect 200 feature point on the first image and track them over all 5 frames. Only 98 features are consistently tracked in all frames and we display all of them in Fig. 5.4c. In parallel we run our segmentation code to obtain an initial grouping of image regions (Fig. 5.4c). The two groups of feature points that reliably produce very similar homographies are displayed in Fig. 5.4e. Based on these homographies we rerun the segmentation code with different parameters until the two regions are merged together (Fig. 5.4d).

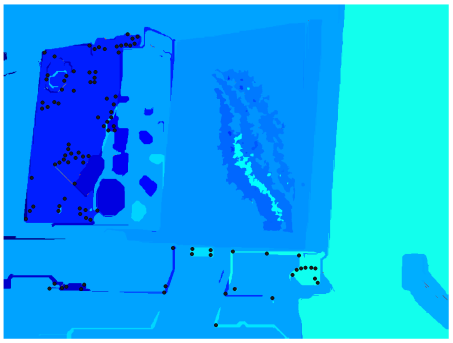
A similar experiment in displayed in Fig. 5.5, where splitting a region into multiple ones is required. As above, we use a KLT tracker to detect 200 feature points and track them over a series of 5 frames. At the end of the process 102 features are tracked (Fig. 5.5c). We also apply our segmentation algorithm with the same parameters (i.e. $h_r = 10$) and obtain the segments shown in Fig. 5.5c. We estimate the homography for each color segment and compute the reprojection

(a) The first image of the sequence.

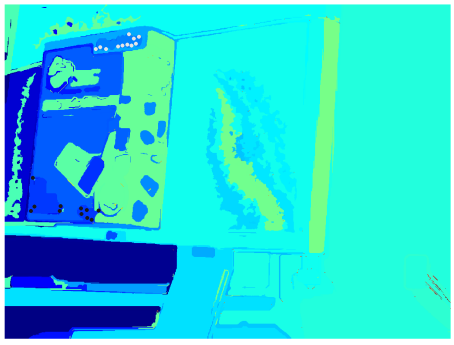
(b) The fifth image of the sequence.



(c) MF+GAT segmentation ($h_r = 10$) with feature points (black dots).



(d) MF+GAT segmentation ($h_r = 30$) with feature points (black dots).

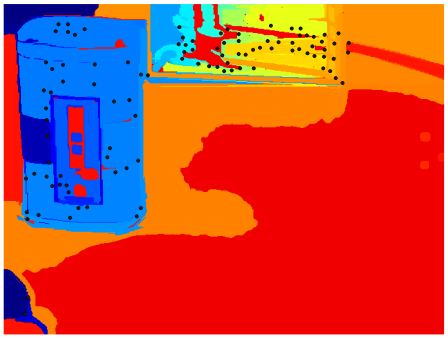


(e) The segments whose feature points are drawn with white and black dots have similar homographies and thus should be merged together.

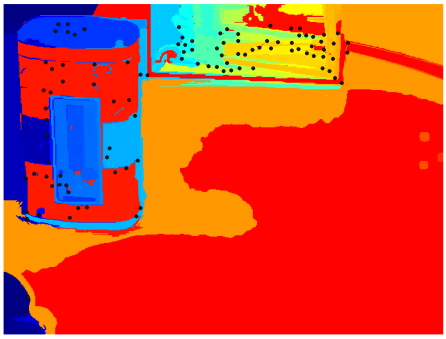
Figure 5.4: An example on how homographies and color segmentation can be combined to obtain better results by region merging.

(a) The first image of the sequence.

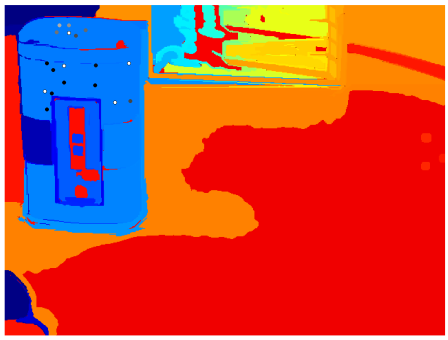
(b) The fifth image of the sequence.



(c) MF+GAT segmentation ($h_r = 10$) with feature points (black dots).



(d) MF+GAT segmentation ($h_r = 7$) with feature points (black dots).



(e) The feature points belonging to the same segment drawn according to their corresponding reprojection error. The brighter the color the higher the error.

Figure 5.5: An example on how homographies and color segmentation can be combined to obtain better results by region splitting.

error of each point. In Fig. 5.5e we display the results for the segment corresponding to the green box. According to our color code the brighter (whiter) a feature the higher the reprojection error for that point is. As expected, the group of features on the top (that belong to a different surface of the box) uniformly exhibit high reprojection errors. The same behavior is exhibited by the features on the right face of the box. This is an indication that we need to further subdivide that region. Fig. 5.5d displays one possible subdivision obtained by running our segmentation algorithm with a different color resolution parameter ($h_r = 7$).

Both examples above should be considered as a proof of concept. We have not reached the point yet, where a segmentation into surfaces is consistently and robustly working in all image sequences. There are some theoretical questions to be addressed and a lot of engineering effort to be made to reach that milestone. For example, it is not clear how to measure the difference between two homographies, or what is the threshold above which a point is considered an outlier. Furthermore, when it comes to our proposed scheme there are additional issues to be addressed. The computation of the color resolution (h_r) to be used for the finer (or coarser) grained segmentation and the verification that the segmentation is the "right" one and thus the procedure should terminate, are two interesting research topics. Moreover, this scheme should be extended to areas where there are not enough feature points. In such cases, methods based on the transformation of the whole region (similar to the warping method proposed in Sec. 4.4) or on the transformation on the boundaries (Sec. 4.3, also see literature about the Iterative Closest Point algorithm [110, 111]) can be used.

5.5 Towards an active approach to image segmentation

We want to conclude this dissertation with a discussion about active vision. We have already showed in chapter 4 that knowledge of the camera motion can facilitate the estimation of surface vectors by turning a non convex, complex optimization problem into a convex linear problem (Sec. 4.5) and by allowing us to create novel algorithms for surface estimation (Sec. 4.4). We believe that the process of surface segmentation can be further simplified and made more accurate by the appropriate camera motion. The homography based segmentation as described on the previous section, and shown in Fig. 5.3, is a passive approach, because the image acquisition process is independent of the segmentation results.

In Fig. 5.6 we modify the approach by connecting the image acquisition module with the segmentation results on a feedback loop. More specifically, the transformation of the passive approach into an active one involves two stages; *prediction* and *optimization*. The former stage incorporates the a priori evaluation of the *expected quality* of the segmentation when the camera performs a specific motion using some objective criterion. Note that as the name suggests, the system should be able to predict the quality without the camera actually performing that motion. The latter stage refers to the process of selecting a camera motion that maximizes the estimated expected quality of the segmentation.

In the context of homography-based plane finding, we argue that the intermediate goal is the accurate estimation of the homographies. Hence, we think that lemma 5.2 is a good starting point to predict the quality of the homography esti-

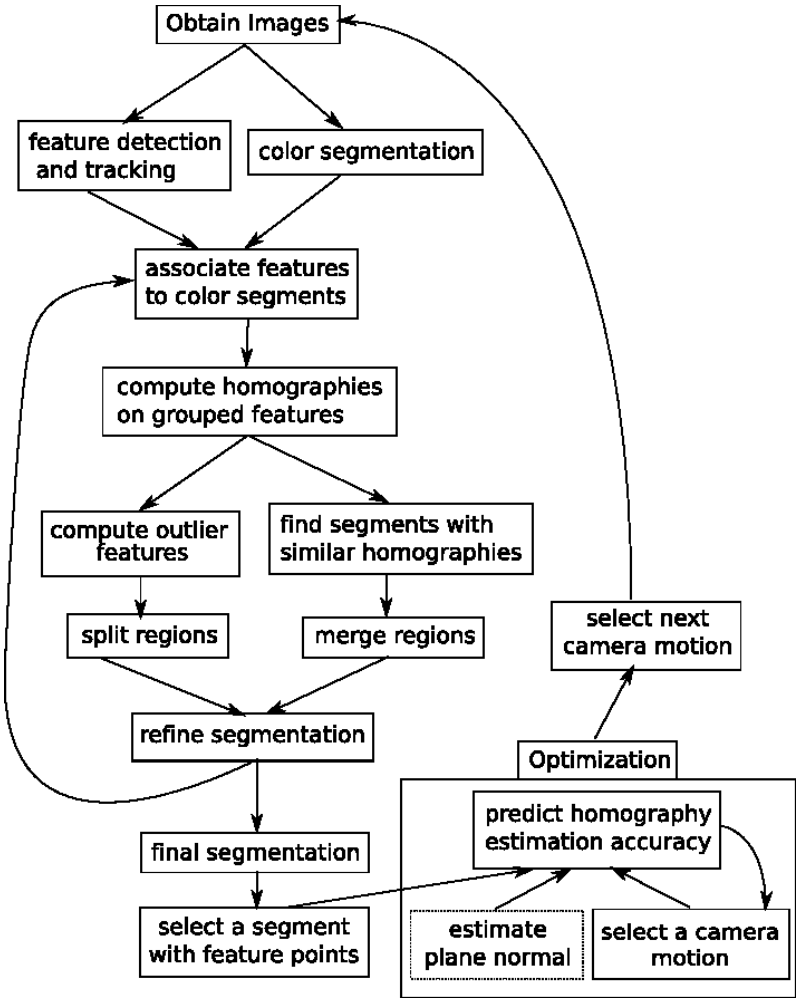


Figure 5.6: Our updated proposed scheme to address the 3D plane estimation problem by combining feature based homography estimation with color based segmentation. A feedback loop for selecting the next best camera position is added, making the whole scheme "active".

mation without actually computing the homographies. In order to build a system, however, a number of related problems needs to be addressed. In our opinion the most important ones are, how to construct the objective function from the covariance matrix of the elements of the homography matrix, and how to find the camera motion that optimizes that objective function.

Appendix A

Segmentation Results for the Weizmann dataset

A.1 The Weizmann Institute dataset

The Weizmann Institute dataset is a new database of images created for the purpose of separating an image into background and foreground regions. As such each image contains a single dominant object, that should be classified as foreground, while the rest of the image is considered as background. In total, the database contains 100 images and 300 human segmentations. There are three significant differences between this dataset and the Berkeley one.

First, all the images are grayscale and not color. As a consequence we perform the filtering on the 3D space (i.e., 2 dimensions for spatial coordinates and 1 dimension for the grayscale intensity values).

The texture variation on these images is significantly less compared to the texture variation on the Berkeley images. This is partially due to the fact that there are fewer images of natural scenes, and mainly because it is harder to encode texture variation on a grayscale image.

In each image only two segments are labeled, the foreground object versus the rest of the image that is considered background. Thus, there are fewer edges labeled in the human segmentations, namely only the edges on the boundary of the foreground object. All the significant edges inside the object as well as the edges of

the background are ignored.

Overall, for all the above reasons, this dataset is less challenging than the Berkeley one. This fact is experimentally proven by the results of the segmentations.

A.2 Experiments

As with the Berkeley dataset we apply all possible combinations of filtering (using the Normal and Epanechnikov kernel) and grouping methods and display the cumulative results for the whole database. To reduce the number of figures we only display the Probabilistic Rand index and the Boundary Displacement Error results. Compared to the parameters we used in the Berkeley dataset we use a much larger range of color resolutions, namely $h_r = 0.5 \ldots 40$ on increments of 0.5. First we present the results when we use CC3D, CC5D and GRAG grouping methods.

The figures with the BDE measure report that Color Mean Shift (CMS) outperforms the other methods not matter if the filtering is performed with a Normal or an Epanechnikov kernel. These plots are similar to the ones for the Berkeley dataset. What come as a surprise are the plots for the Probabilistic Rand index. Not only they show that the segmentations become better as the average segment size increases, asymptotically reaching the value of 1 (which is ideal), they also present the methods that performed poorly on the previous dataset, e.g. Local Mode filtering with GRAG grouping, (and on the current dataset considering the BDE measure) to outperform all the other methods. After checking the results for the individual images we realized that the suspiciously good values for the PR index

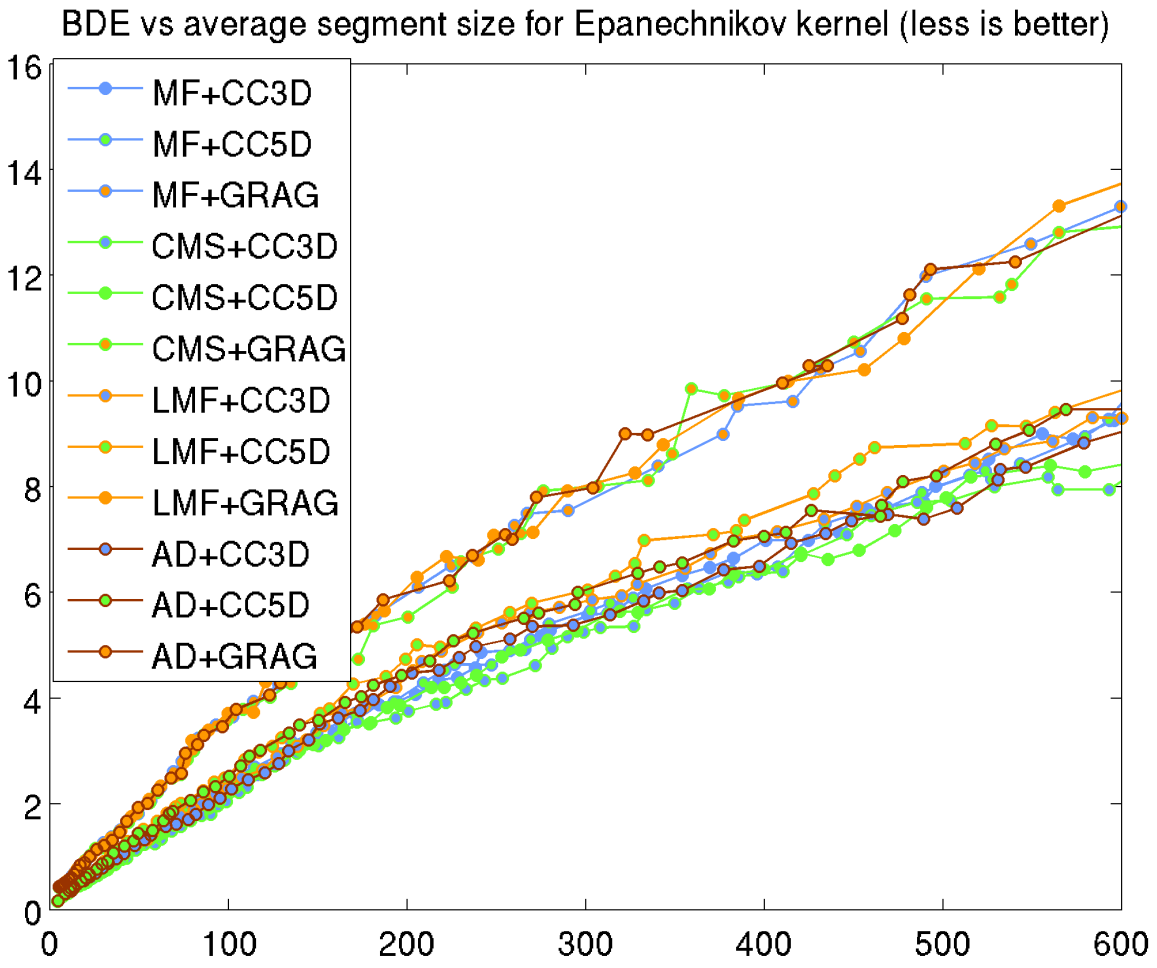


Figure A.1: BDE vs average segment size plots for the Weizmann dataset when filtering is performed with an Epanechnikov kernel.

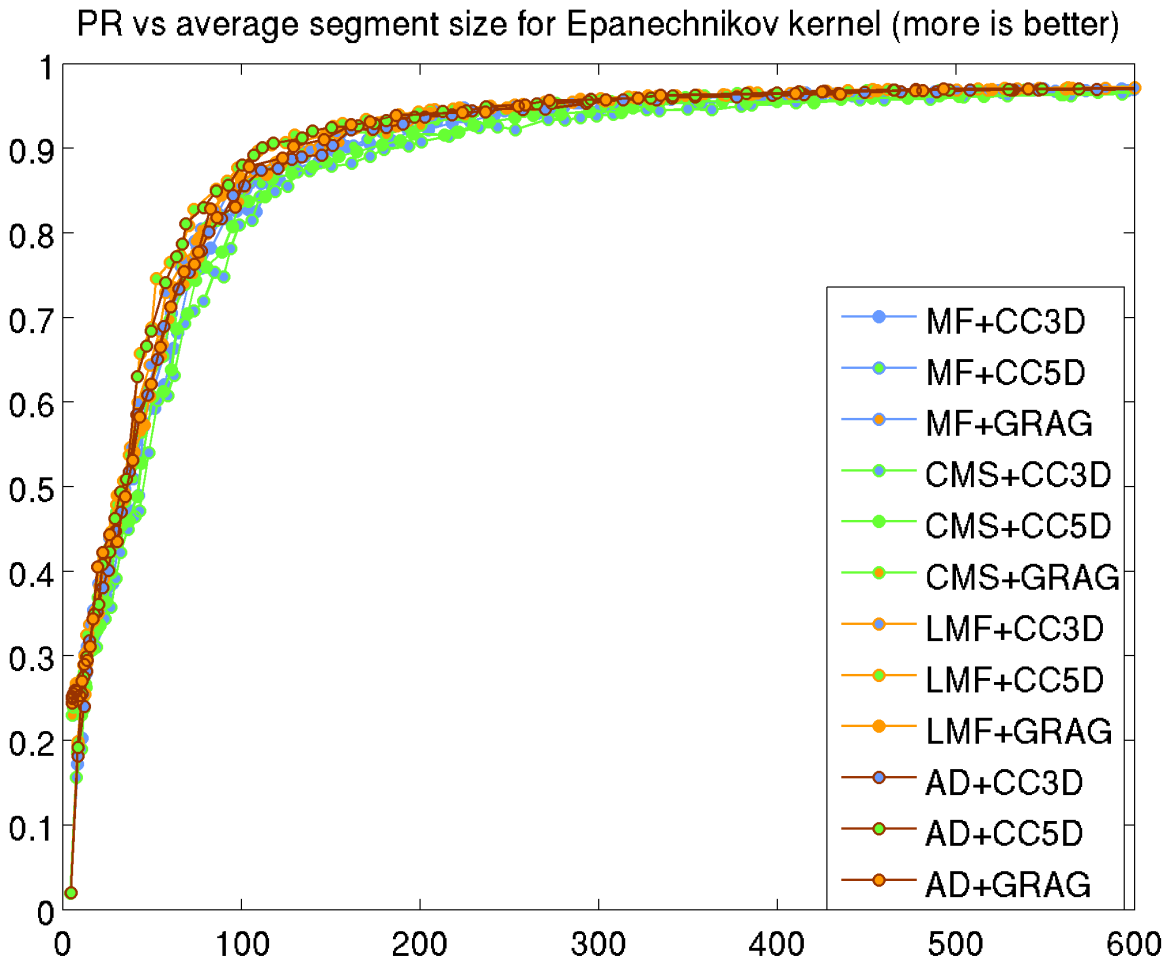


Figure A.2: PR vs average segment size plots for the Weizmann dataset when filtering is performed with an Epanechnikov kernel.

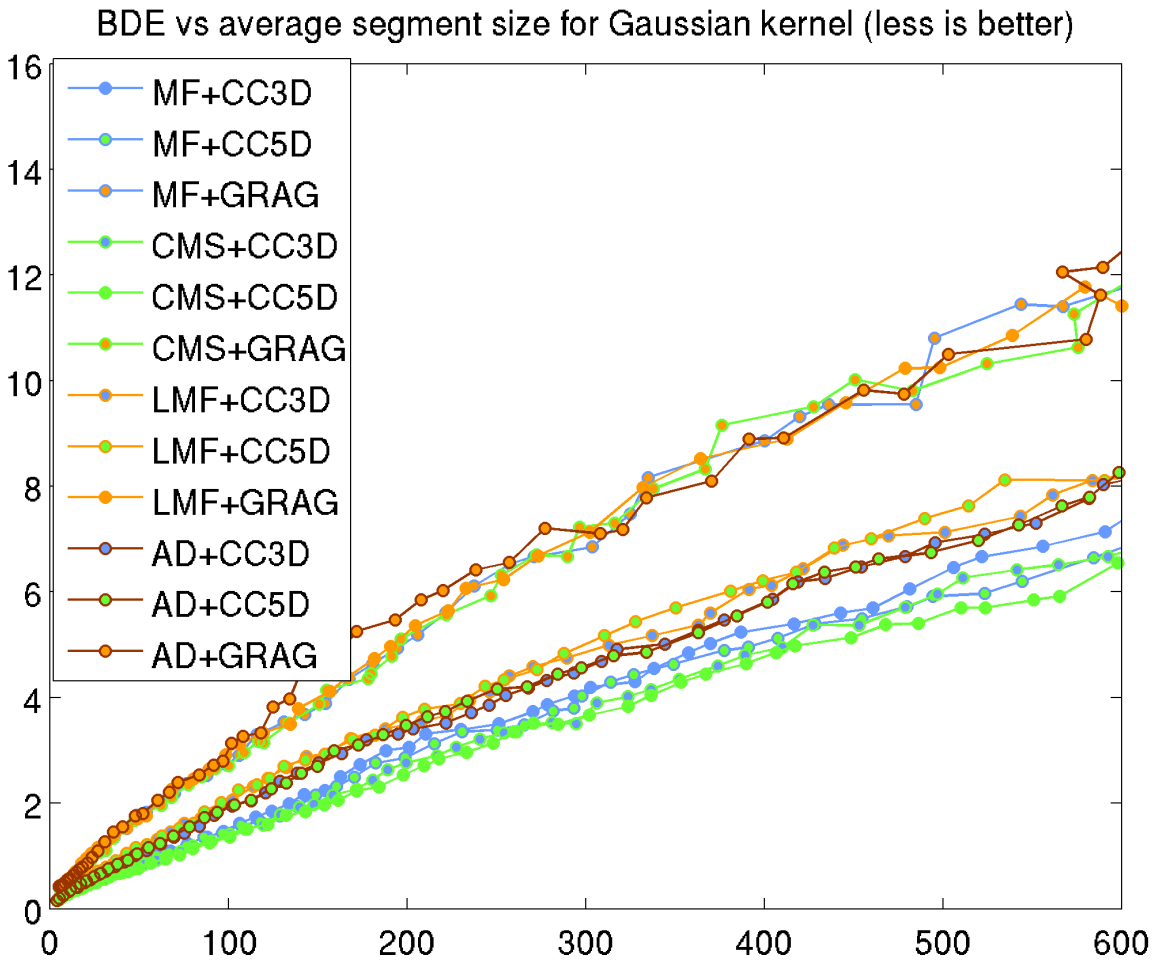


Figure A.3: BDE vs average segment size plots for the Weizmann dataset when filtering is performed with a Normal kernel.

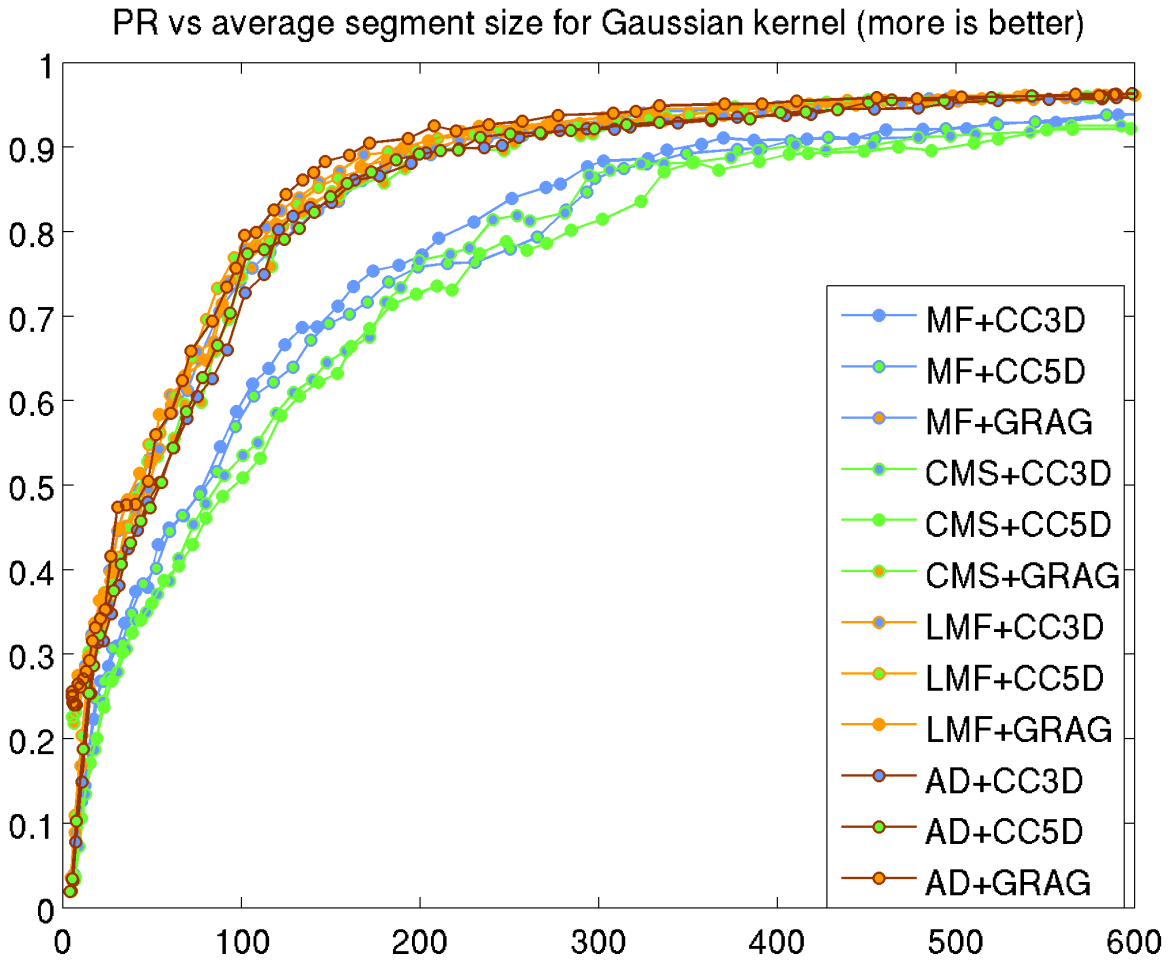
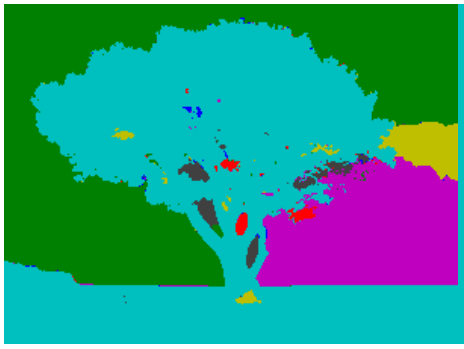


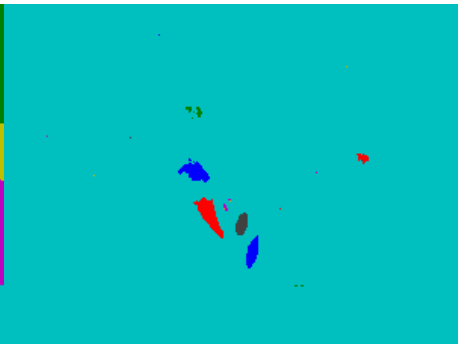
Figure A.4: PR vs average segment size plots for the Weizmann dataset when filtering is performed with a Normal kernel.

(a) Original Image

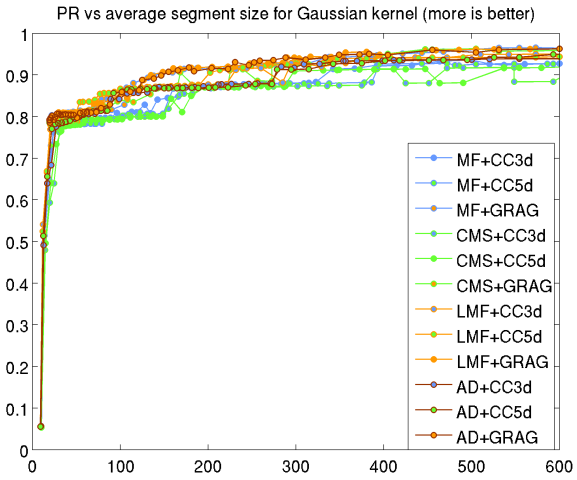
(b) Human Segmentation



(c) MF+CC3D ($h_r = 40$), PR=0.37



(d) MF+CC3D ($h_r = 80$), PR=0.95



(e) PR measure vs average image size

Figure A.5: A single image of the Weizmann dataset along with the human and computer generated segmentation for two different color resolutions ($h_r = 40,\ 80$) and their corresponding Probabilistic Rand values (0.37, 0.95). Notice that the second segmentation produces a much higher PR value even if it is much worse than the first segmentation. This is a problem of the PR measure when applied to foreground/background segmentation images. In this example a segmentation of the whole image into a single region produces a PR value of 0.97.

are due to the nature of the human segmentations. More specifically, with only one foreground and one background region, the PR index for a segmentation of the whole image into a single region is very high. For example Fig. A.5 shows the results for the first image of the database and the first segmentation. One reasonable computer segmentation (Fig. A.5c) has the PR value of 0.37. The much worse segmentation of Fig. A.5d produces a PR value of 0.95. Finally, the segmentation into a single segment produces a PR index of 0.97, that is very close to the absolute best value of 1. Thus, we conclude that *PR is not a good index of the quality of the segmentation for that specific database.*

Notice that this problem was not present in the Berkeley dataset. Looking at any of the PR figures (e.g. Fig. 3.7), one sees the parabolic like shape of all the plots that indicates that for both very small and very large segment sizes the PR values are bad (as expected).

In Fig. A.6 we plot on the same graph the segmentation results of the two prominent filtering methods (i.e., Mode Finding and Color Mean Shift) using Epanechnikov and Normal kernels coupled with the CC3D grouping method. Comparing the segmentation results along regular average segment size intervals (i.e. from ~ 100 pixels to ~ 800 pixels in intervals of 50) we get the following numbers. For the Color Mean Shift method the Normal kernel results are on average $\sim 31\%$ better than the Epanechnikov results, while Mode Finding with a Normal kernel produced $\sim 33\%$ better results on average than the Epanechnikov based method.

Using the same numbers we were able to quantify how better the segmentation results for Color Mean Shift were compared to the Mode Finding ones. When using

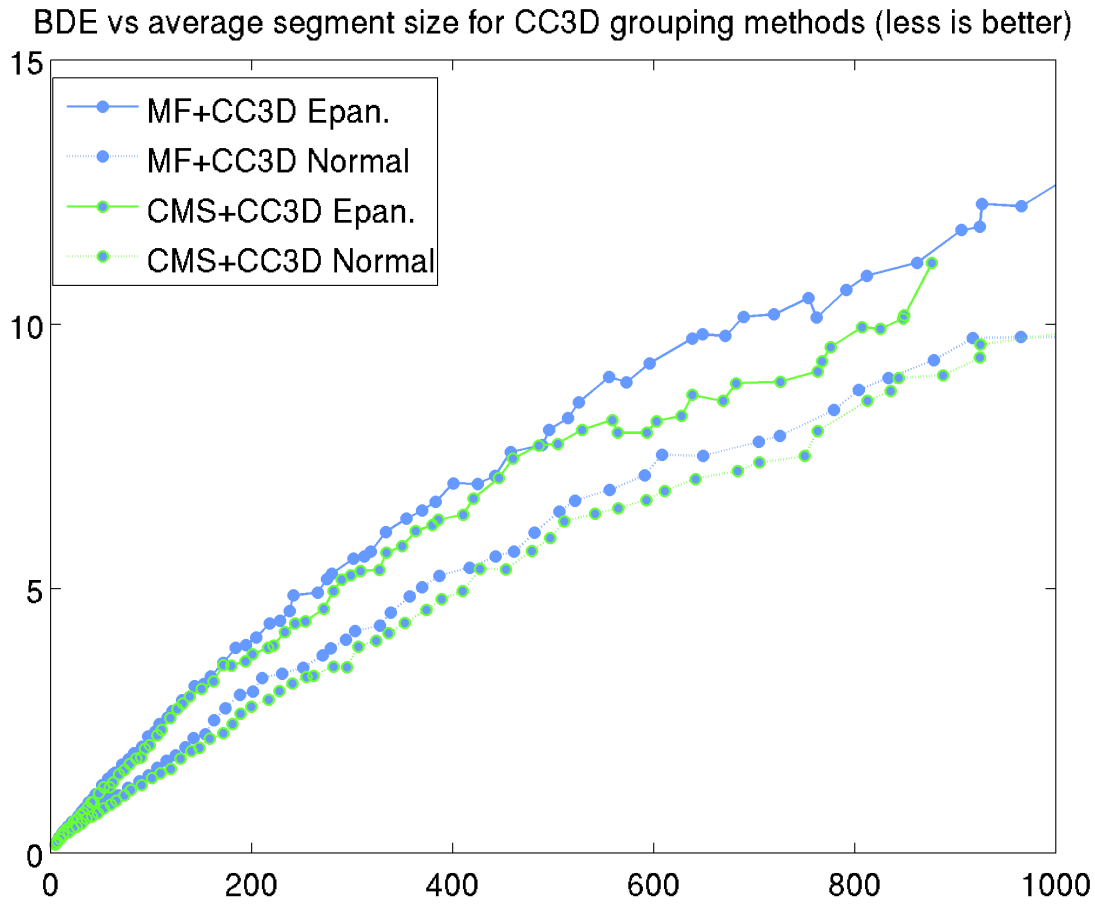


Figure A.6: BDE vs average segment size plots for the Weizmann dataset. The goal is to compare the performance when we use different filtering kernels. In this plots we use the Mode Finding and Color Mean Shift filtering methods along with the CC3D grouping methods. The dotted and solid line plots denote filtering with a Normal and Epanechnikov kernel respectively. The Normal kernel clearly outperforms the Epanechnikov kernel. Similar results were obtained with different combinations of filtering and grouping methods.

a Normal kernel CMS produced $\sim 8\%$ better results, while for the Epanechnikov kernel the value was slightly higher at $\sim 9.4\%$ compared to Mode Finding. We observed similar increases in the performance when we used the other grouping methods as well.

Next we move our attention to the grouping method with an adaptive threshold (GAT). In Sec. 3.4.3 we presented one way to linearly adjust the segmentation parameter k according to the filtering parameter h_r (Eq. 3.1). Figs. A.7, A.8 and A.9 present the segmentation results of applying this method to the Weizmann Institute dataset. More specifically, Figs. A.7, A.8 present the results when the filtering is done with an Epanechnikov and a Normal kernel respectively, while in Fig. A.9 we compare the best performing methods.

The results for all the filtering methods with an Epanechnikov kernel are consistently better compared to using the GAT method without any filtering. In the cases of filtering with a Normal kernel, for a range of average segment sizes up to ~ 150 pixels all the methods outperform the non filtering alternative. For larger average segment sizes Color Mean Shift and Anisotropic Diffusion perform much worse that GAT only, while Mode Finding and Local Mode Filtering performs equally well. Overall, GAT coupled with Local Mode Filtering with a Normal kernel seems to perform best for segment sizes up to ~ 150 pixels and Mode Finding with an Epanechnikov kernel is the best performing method for larger segment sizes.

On the last figure (Fig. A.10) of this appendix we compare the best segmentation methods using CC3D and GAT with varying k for the grouping step. It is clear that grouping with an adaptive threshold outperforms the simple connected compo-

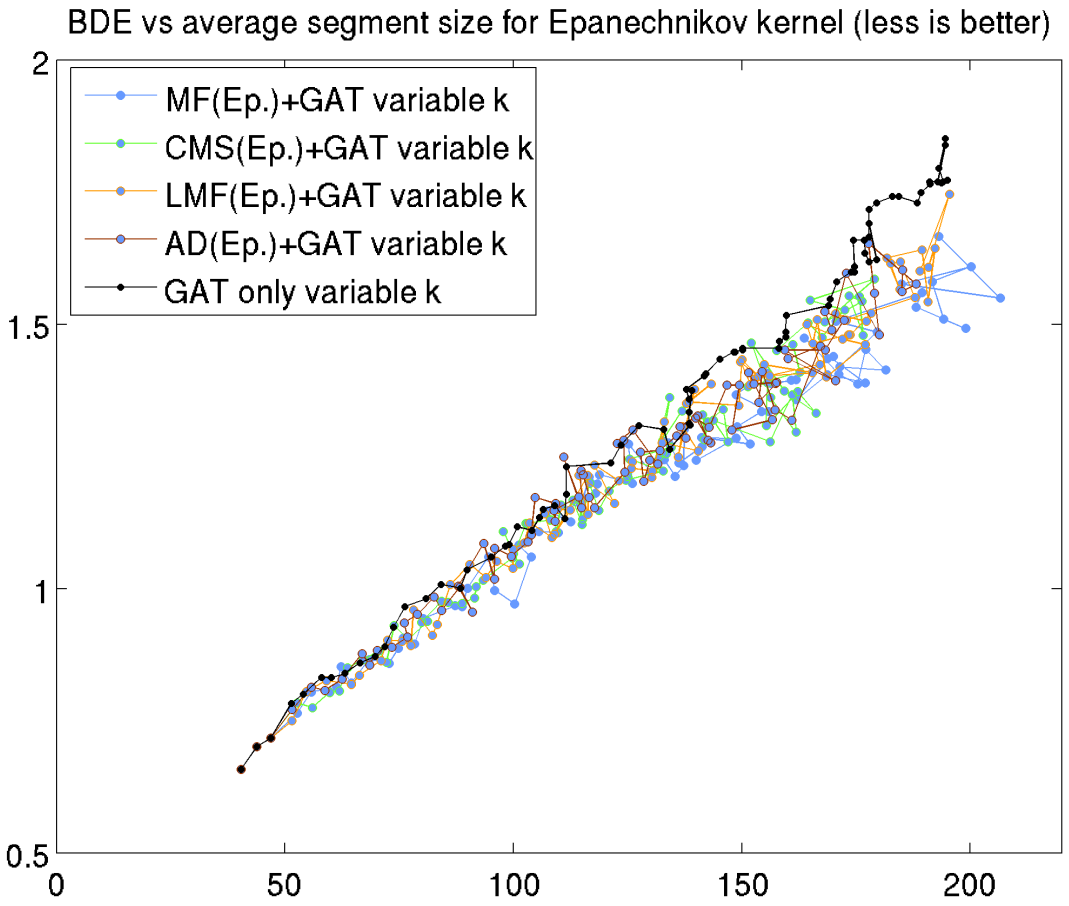


Figure A.7: BDE vs average segment size plot for the GAT grouping method with variable k preceeded by the filtering methods with an Epanechnikov kernel. We also display the plot when we use the GAT grouping method without any filtering method (back curve).

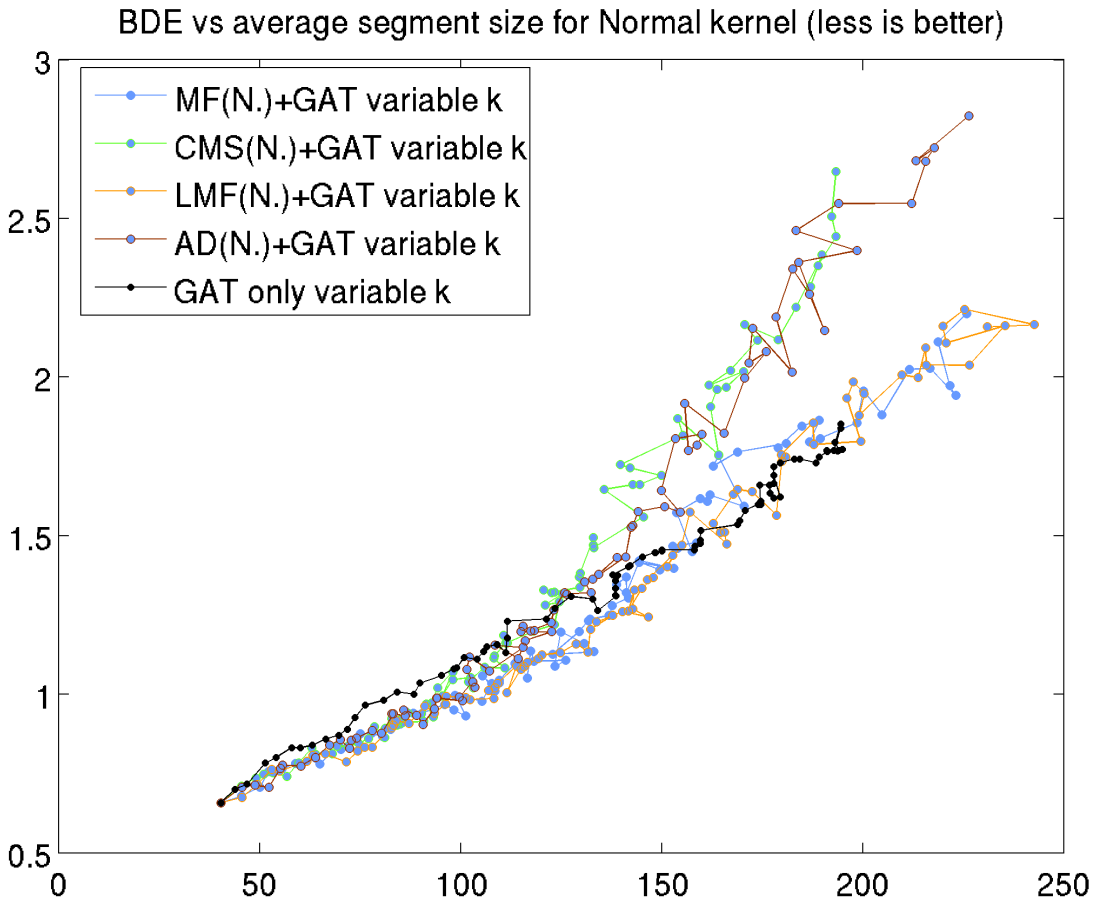


Figure A.8: BDE vs average segment size plot for the GAT grouping method with variable k preceeded by the filtering methods with a Normal kernel. We also display the plot when we use the GAT grouping method without any filtering method (back curve).

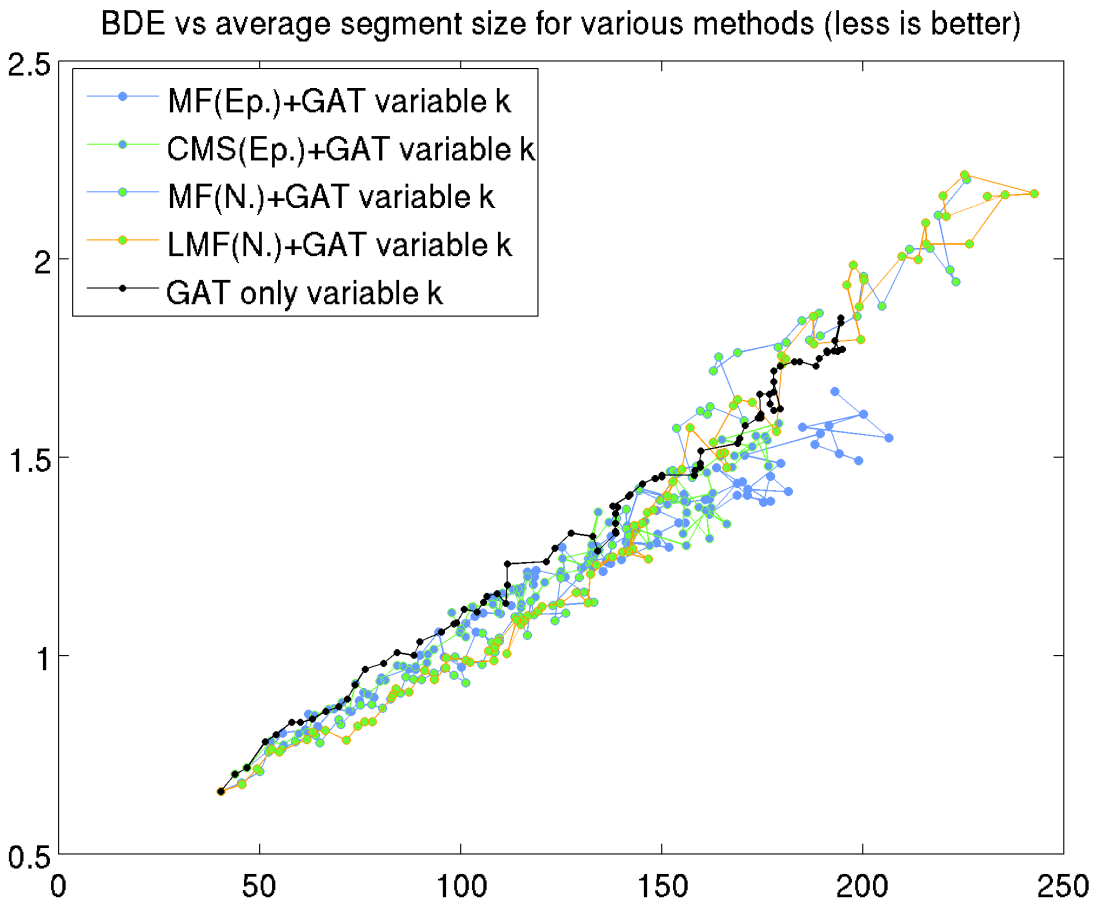


Figure A.9: BDE vs average segment size plot for the GAT grouping method with variable k preceeded by selected filtering methods. We also display the plot when we use the GAT grouping method without any filtering method (back curve).

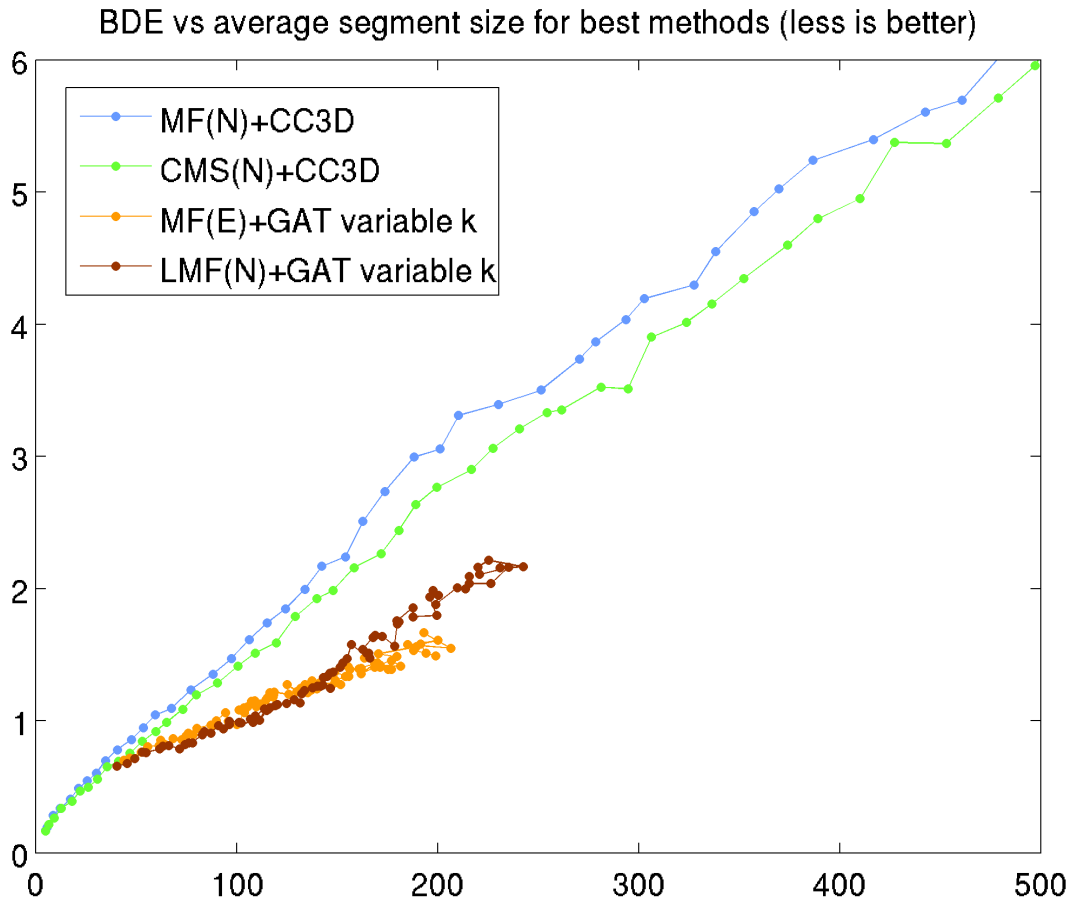


Figure A.10: BDE vs average segment size plot for the best methods using a hard and an adaptive threshold for grouping. GAT based grouping methods outperform the CC3D based methods.

Table A.1: Conclusions summary

- CMS outperforms all the other filtering methods when grouped with CC3D, CC5D or GRAG (Figs. A.2, A.4, A.6).
- Normal kernel outperforms Epanechnikov kernel filtering (Figs. A.6).
- Segmentation methods based on filtering and grouping outperform methods based on grouping only (Figs. A.7, A.8).
- MF and LMF filtering combined with GAT with varying k grouping perform better than CMS and AD filtering (Figs. A.7, A.8).
- GAT based methods outperform CC3D, CC5D and GRAG based methods (Fig. A.10).

nents based grouping methods. All the result of the experiments on the Weizmann Institute dataset are summarized in Table A.1.

Appendix B

Stretch Filter

B.1 Introduction

We proved in a previous chapter that the surface normal parameters, namely the slant of a plane, is encoded in the stretch between two epipolar lines (Eq. 4.12). This is one example of a general problem where, one is given two signals, one of them being a transformed version of the other, and the goal is to recover this transformation. As we described earlier (Alg. 4.2), assuming one wants to estimate the zero (shift) and first order (stretch) component of the transformation, a general method is to use the log of the magnitude of the Fourier transform. This technique, which is known as *Cepstral analysis*, was first introduced by Bogert et al. [59] and was made widely known by Oppenheim and Schafer [60]. It is commonly used in speech processing [61] to separate different parts of the speech signal. Cepstral analysis requires an explicit FTT on both signals with complexity $O(N \log(N))$.

Phase-difference based techniques exploiting the phase shift theorem have also been used in computer vision. In most cases the assumption is that one signal is simply shifted relative to another, thus only the zero order component is estimated. In this case, a very robust way to recover the amount of shift is through measurements of the change of phase in different frequencies. In a classic paper on stereo, Sanger [112] convolved each scan line of two images with Gabor filters. Similar work at the

same time period was performed by Jenkin and Jepson [113]. Fleet and Jepson in [114] and [115] studied the stability of the previous techniques in the presence of phase singularities and identified patterns on the phase domain where the previous methods fail.

For $2D$ signals, Srinivasa et al. [62] recover the *global* relative translation, uniform scale and image rotation (i.e., 4 parameters) of two images by analyzing the changes in various Fourier components. It is straightforward to use the Fourier Shift property to recover translations. However, they pursue a different strategy to recover different components. Specifically, they show that by performing a log-polar transformation on each image, rotation and scaling can be transformed into translations. *Local* phase-based techniques have also been developed for optical flow estimation. Fleet and Jepson's method [65] use Gabor filters to locally compute the phase of two $2D$ signals, and estimate the local shift (i.e., optical flow) of the two signals. While, their technique is shown to outperform most other methods in terms of accuracy and robustness ([116]), it still assumes that all the components of the transformation higher than zero order are zero.

Recent stereo approaches recognize the importance of higher order components of the transformation and try to estimate them. For example, Ogale and Aloimonos in [117] attempt to recover both the shift and the stretch of the transformation by trying many possible warpings of the image, in order to compensate for the stretch component, and choosing the one leading to best matches.

In this chapter, we present an approach similar to Sanger's, but instead of measuring the translation, we directly recover the "stretch" (a linear factor) of two

signals. Related to our approach are the “scale representation" by L. Cohen [118] and the Mellin transform. Both of these methods decompose the signal using a set of basis functions. The stretch is encoded as a phase shift in these representations. Conversely, our method uses only a single filter to estimate the stretch. More explicitly, our main contributions are:

- We analytically create a filter that is able to directly measure the local stretch of two signals (Sec. B.3).
- We present experimental results where we apply this filter to shifted and stretched real signals (Sec. B.4).

Overall our method is much faster then the other approaches, since it only requires the application of a single filter at one point in each image. This computational advantage is offset to an increased sensitivity to errors in shift estimation.

B.2 Gabor Function and notation preliminaries

According to its definition, a Gabor filter consists of a Gaussian function of spatial bandwidth σ, that modulates a complex sinusoid of frequency ω.

$$G(x, \omega, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} e^{2\pi i \omega x} \tag{B.1}$$

We consider the spatial bandwidth (σ) to be fixed with respect to the frequency (ω)

$$\sigma = \frac{c}{\omega}, \tag{B.2}$$

where c is a constant (e.g. Sanger uses $c = 1$ [112]) . As a consequence, the Gabor function only has two parameters, namely x and ω.

We use a calligraphic font for the Fourier transform ($\mathcal{F}_\omega$) of a signal (or a filter). In order to avoid any confusion, we denote with a subscript the integration variable when needed.

B.3 Estimating the stretch

Suppose that one is given two signals $I_1(x)$ and $I_2(x)$, where I_2 is a "stretched version of I_1.

$$\forall x \in \mathbb{R},\ I_2(x) = I_1(\alpha x) \tag{B.3}$$

In this paper we describe a way to estimate the unknown stretch parameter α. Our approach is based on two observations:

- Convolving the first signal (I_1) with a Gabor filter of frequency ω is equivalent to convolving the second signal (I_2) with a Gabor filter of frequency $\alpha\omega$ (Theorem B.1).

- Considering the log-frequency domain of the Gabor filters the multiplication is transformed into addition (i.e., stretch is transformed into shift) and thus can be estimated using the phase shift property of the Fourier transform (Theorem B.2).

In the remaining section we formally present our approach in incremental steps using two theorems. Note that the final result is a *single filter on the spatial domain*, even

thought we are using the frequency domain in our proofs.

Theorem B.1. *If the two stretched signals (I_1, I_2) are as in Eq. B.3, then*

$$\forall\omega,\ [I_1(x) \star G(x,\omega)](0) = [I_2(x) \star G(x,\alpha\omega)](0) \tag{B.4}$$

Proof. According to the definition of a Gabor filter (Eq. B.1) and its standard deviation (Eq. B.2) we get

$$G(x,\alpha\omega) = \frac{1}{\sqrt{2\pi}\sigma'} e^{-\frac{x^2}{2\sigma'^2}} e^{2\pi i\alpha\omega x},$$

$$\sigma' = \sigma_{\alpha\omega} = \frac{c}{\alpha\omega} = \frac{\sigma_\omega}{\alpha}$$

Thus,

$$G(x,\alpha\omega) = \frac{\alpha}{\sqrt{2\pi}\sigma} e^{-\frac{x^2\alpha^2}{2\sigma^2}} e^{2\pi i\omega\alpha x} = \alpha G(\alpha x,\omega). \tag{B.5}$$

From the definition of convolution we have

$$[I_1(x) \star G(x,\omega)](0) = \int_x I_1(x) G(-x,\omega) dx. \tag{B.6}$$

Similarly,

$$\begin{aligned}[I_2(x) \star G(x,\alpha\omega)](0) &= \int_x I_2(x) G(-x,\alpha\omega) dx \\ &= \int_x I_1(\alpha x) G(-x,\alpha\omega) dx \end{aligned} \tag{B.7}$$

Setting $y = \alpha x$, then $dy = \alpha dx$,

$$[I_2(x) \star G(x, \alpha\omega)](0) = \int_y I_1(y) G(-\frac{y}{\alpha}, \alpha\omega) \frac{dy}{\alpha}.$$

Using Eq.B.5 we have

$$\begin{aligned}[I_2(x) \star G(x, \alpha\omega)](0) &= \int_y I_1(y) G(-y, \omega) dy \\ &= [I_1(x) \star G(x, \omega)](0).\end{aligned}$$

□

Based on Theorem B.1 the response of the convolution of I_1, I_2 with the Gabor filter is a function of the frequency ω, that is

$$R_1(\omega) = [I_1(x) \star G(x, \omega)](0) =$$

$$[I_2(x) \star G(x, \alpha\omega)](0) = R_2(\alpha\omega). \tag{B.8}$$

If we consider the log frequency ψ instead of the frequency ω

$$\psi = e^{\omega} \Leftrightarrow \omega = \log \psi, \tag{B.9}$$

then Eq. B.8 is transformed to

$$R_1(\psi) = R_2(\psi + \log \alpha). \tag{B.10}$$

In principle, we could estimate the shift (in the log-frequency domain ψ) by transforming it into a phase shift using the Fourier transform

$$\mathcal{R}_1(u) = \mathcal{F}_\psi\{R_1\} = e^{2\pi i \log \alpha u}\mathcal{R}_2(u) \tag{B.11}$$

and measuring the difference in the phase of $\mathcal{R}_1$ and $\mathcal{R}_2$ for any specific frequency u[1]. While this is a valid approach, it is rather computationally expensive. For every point of the two signals one has to compute the frequency response R_1, R_2 (by convolving with Gabor filters of different frequencies) and then take the Fourier transform of those responses. The following theorem provides an alternative solution that amounts to convolving the two signals with a single filter.

Theorem B.2. *There exist filters $H(x,u)$ whose convolution with I_1, I_2 directly encodes the stretch as*

$$[I_1(x) \star H(x,u)](0) = e^{2\pi i \log \alpha u}[I_2(x) \star H(x,u)](0). \tag{B.12}$$

Specifically, the filters have the analytic form

$$\boxed{H(x,u) = \int_\omega G(x, e^\omega) e^{-2\pi i \omega u} d\omega,} \tag{B.13}$$

where u, ω are free parameters that define the form of the filter.

[1]We have noticed that the following issue is often at first confusing to readers. We use two different frequency domains. Symbols ω (and ψ) denote the frequency in the "traditional" sense, while symbol u denotes the Fourier transform of ψ, so in some sense is the *"frequency of the frequency domain"*.

Proof. From Eqs. B.8, B.9 and B.10 we have

$$R_1(e^{\omega}) = R_2(e^{\omega} + \log \alpha)$$

If we consider the Fourier transform of $R_1(e^{\omega})$ with respect to ω, then

$$\begin{aligned}
\mathcal{R}_1(u) &= \int_{\omega} e^{-2\pi i \omega u} R_1(e^{\omega}) d\omega \\
&= \int_{\omega} e^{-2\pi i \omega u} [\int_{x} I_1(x) G(-x, e^{\omega}) dx] d\omega \\
&= \int_{x} I_1(x) [\int_{\omega} G(-x, e^{\omega}) e^{-2\pi i \omega u} d\omega] dx \\
&= \int_{x} I_1(x) H(-x, u) dx \\
&= [I_1(x) \star H(x, u)](0).
\end{aligned}$$

Similarly for $R_2(e^{\omega})$ we get

$$\mathcal{R}_2(u) = [I_2(x) \star H(x, u)](0).$$

From the phase shift property of the Fourier transform we get

$$\mathcal{R}_1(u) = e^{2\pi i \log \alpha u} \mathcal{R}_2(u)$$

and thus

$$[I_1(x) \star H(x, u)](0) = e^{2\pi i \log \alpha u} [I_2(x) \star H(x, u)](0).$$

□

Algorithm B.1 Stretch estimation with a single filter.

Input	:	
I_1, I_2	:	Input Signals
x_0	:	A single point along the X axis
Output		
α	:	The stretch between the two signals around point x_0

Algorithm

Create the filter $H(x,u) = \int_{\omega_1}^{\omega_2} G(x, e^{\omega}) e^{-2\pi i \omega u} d\omega$
Convolve the two signals I_1, I_2 with $H(x,u)$ around x_0
Compute the difference in phase of the two measurements $\Delta\theta$
Compute the "log-frequency shift" $\Delta\psi = \frac{\Delta\theta}{2\pi u}$
Compute the stretch $\alpha = e^{\Delta\psi}$

The algorithm is a straightforward implementation of the theory and is presented in Alg. B.1. Notice that we use a bounded integral in order to estimate the filter H, with lower and upper bounds on the frequency variable ω_1, ω_2, respectively. Also notice that we precompute *a single filter* and we use the same filter in both signals I_1, I_2.

The computation of stretch around a point x_0 involves the convolution of the two signals, a computation of a phase difference, a division and an exponentiation, thus if the size of the filter is M the complexity of the algorithm is $O(M)$.

B.3.1 How Parameter u affects H

Symbol u is used to denote the frequency domain produced by considering the Fourier transform of the Gabor filters with respect to their frequencies (ω). Thus, intuitively a filter H with small u frequency is "smoother" than one with high u. Fig. B.1 shows two different filters for $u = 0.25$ and $u = 1$.

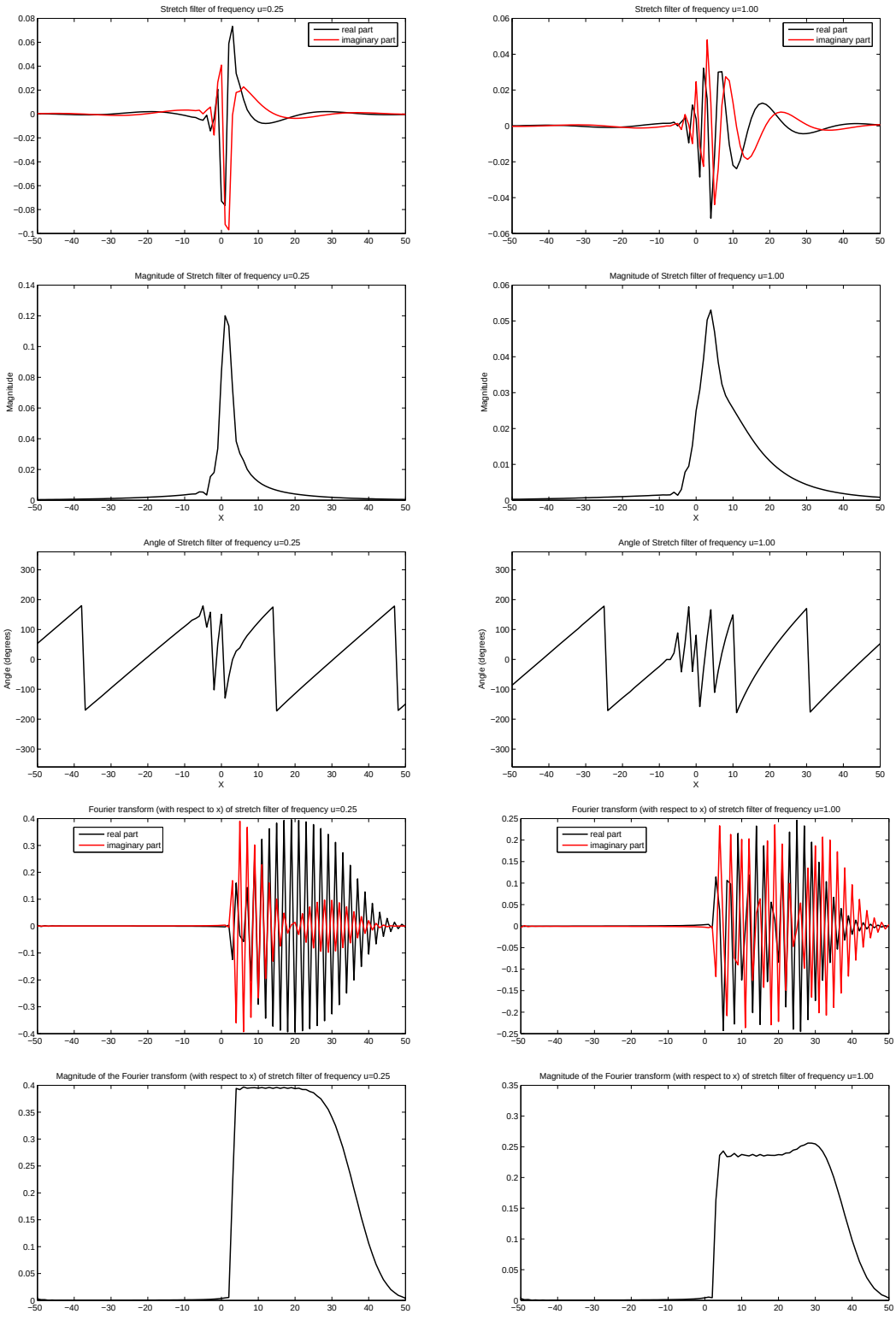


Figure B.1: Dependence of $H(x,u)$ on u. In the first column $H(x,0.25)$ is displayed while in the second column $H(x,1)$. The first filter is "smoother" than the second one.

B.4 Experiments

B.4.1 Stretch without Shift Experiments

On this first set of experiments, the original signal (I_1) is the horizontal lines of various textures [119] (Fig. B.2). We randomly selected 200 scan lines and stretched each one of them around its center in order to produce a second signal (Fig. B.3, first and second row). Then we convolved both signals with a *single filter* of frequency $u = 0.25$ as shown in Fig. B.3 (third row). Following the steps described in Alg. B.1 we estimated the stretch. We experimentally found that frequencies in the range $u = [0.25 \ldots 0.5]$ worked well. The higher the frequency, the better the results for stretches closer to 1 and the worse for stretches closer to 0. For the lower and upper bounds of integral H (Eq. B.13) we used the values -3.5 and -1, respectively.

In Fig. B.4 we present the results as a function of the stretch α. Each graph corresponds to an image from Fig. B.2. For each stretch value we pick 200 random points and synthetically stretch the signal about each. We plot both the *median* value and the 99% *confidence interval* for the estimated stretches. The results are good considering the following facts. First, we are using a single filter to estimate the stretch. Second, the size of the filter is $\sim$ 20 pixels. Third, we have discrete signals, thus for a stretch of $\alpha = 0.5$ only 10 pixels are common in the original and the stretched image. Fourth, for practical purposes, we are usually interested in stretches close to one (e.g. $\alpha = [0.9 \ldots 1.1]$) in which case the estimated stretch is quite accurate. Thus, in Fig. B.5 we display the results on that range of stretches. In all cases the estimated stretch is very close to the real stretch between the two

(a) Cross tiles

(b) Roman tiles

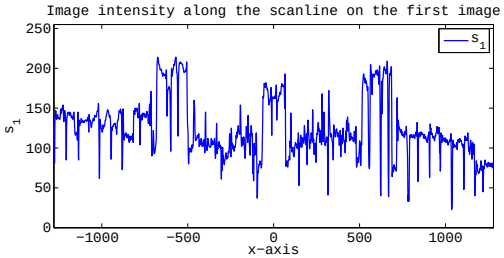
(c) Peddles

(d) Brick wall

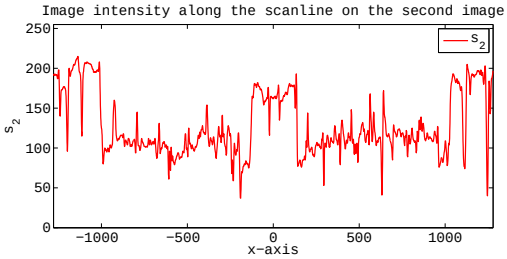
Figure B.2: Texture images for stretch experiments.

(a) Original image

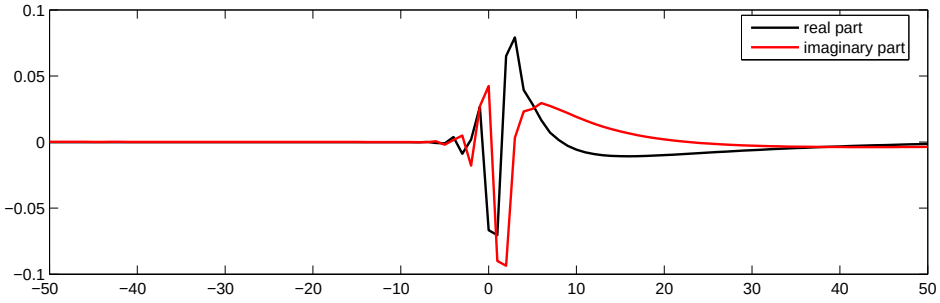
(b) Stretched image ($\alpha = 0.5$)



(c) $1D$ intensity signal along a scan line

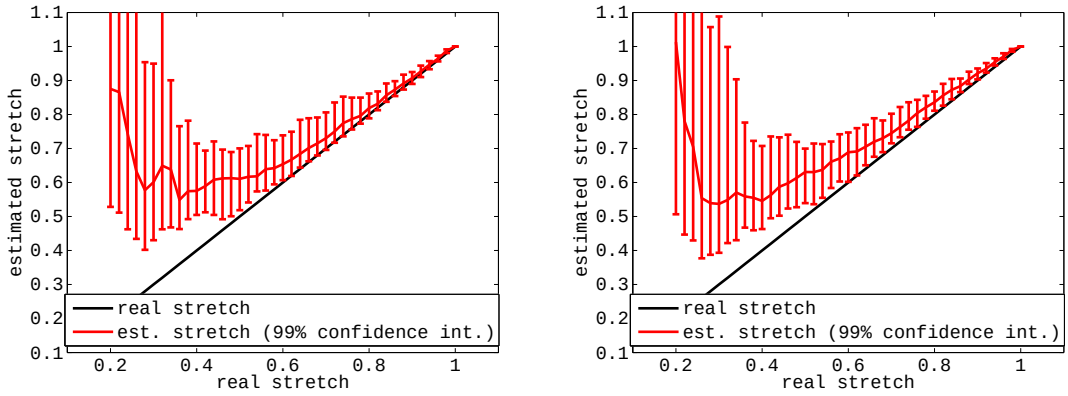


(d) $1D$ stretch signal along the same scan line



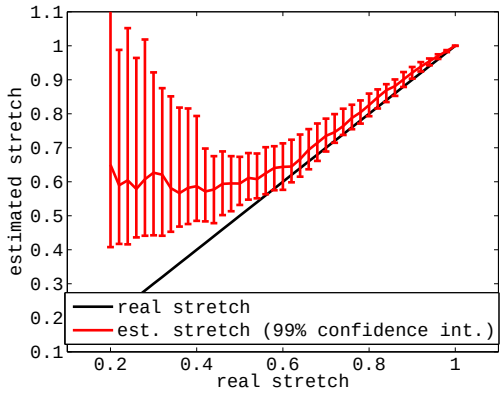
(e) The stretch filter H we use

Figure B.3: An example of an original and stretched image along a scanline, along with the filter H that is used throughout the experiments.

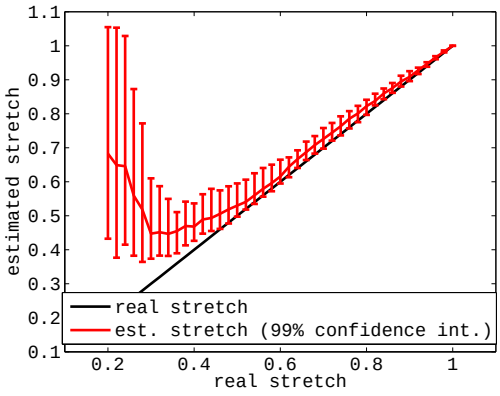


(a) Cross tiles

(b) Roman tiles

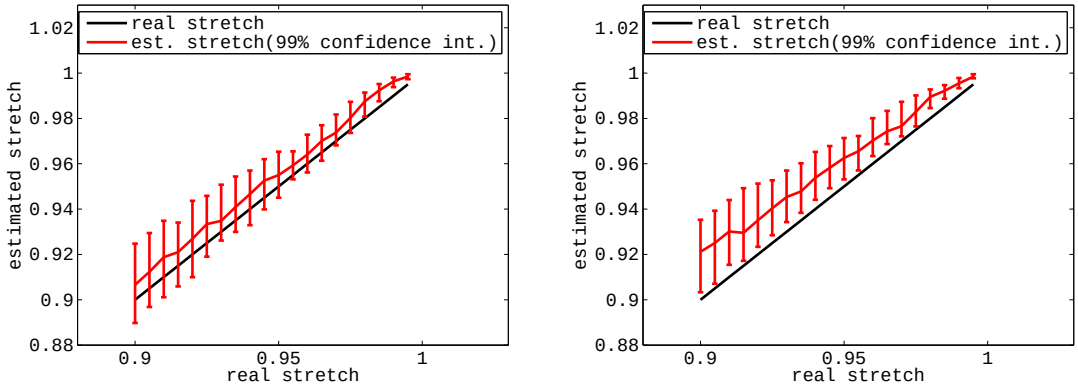


(c) Peddles

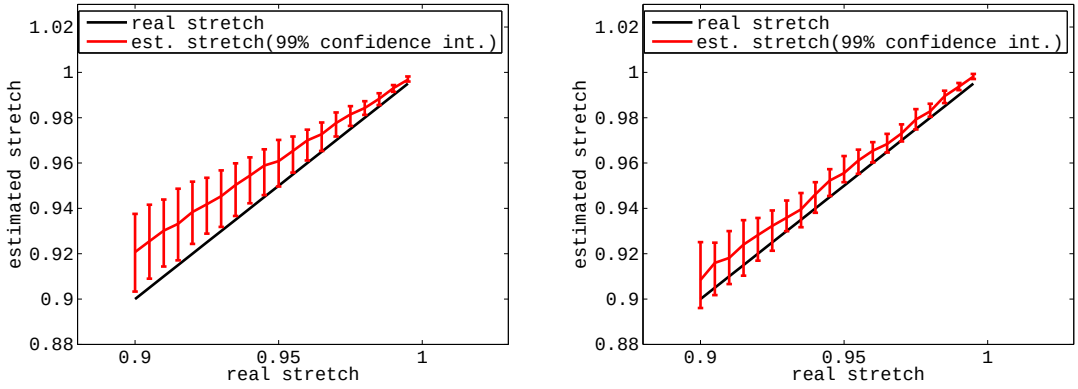


(d) Brick wall

Figure B.4: The estimation results for various amounts of stretch of the image.



(a) Cross tiles (b) Roman tiles



(c) Peddles (d) Brick wall

Figure B.5: Estimation results for various amounts of stretch when α is close to 1.

signals.

B.4.2 Stretch Estimation in the Presence of Translation

In real applications, the most common case is for the two signals to be both shifted and stretched i.e., $i_2(x) = i_1(\alpha x + \beta)$. In such cases, estimation of the stretch (α) is affected by the shift (β) and vice versa. In the following experiments, we empirically investigate the sensitivity of the stretch estimation in the presence of translation between the two signals. Fig. B.6 we display the error in the stretch

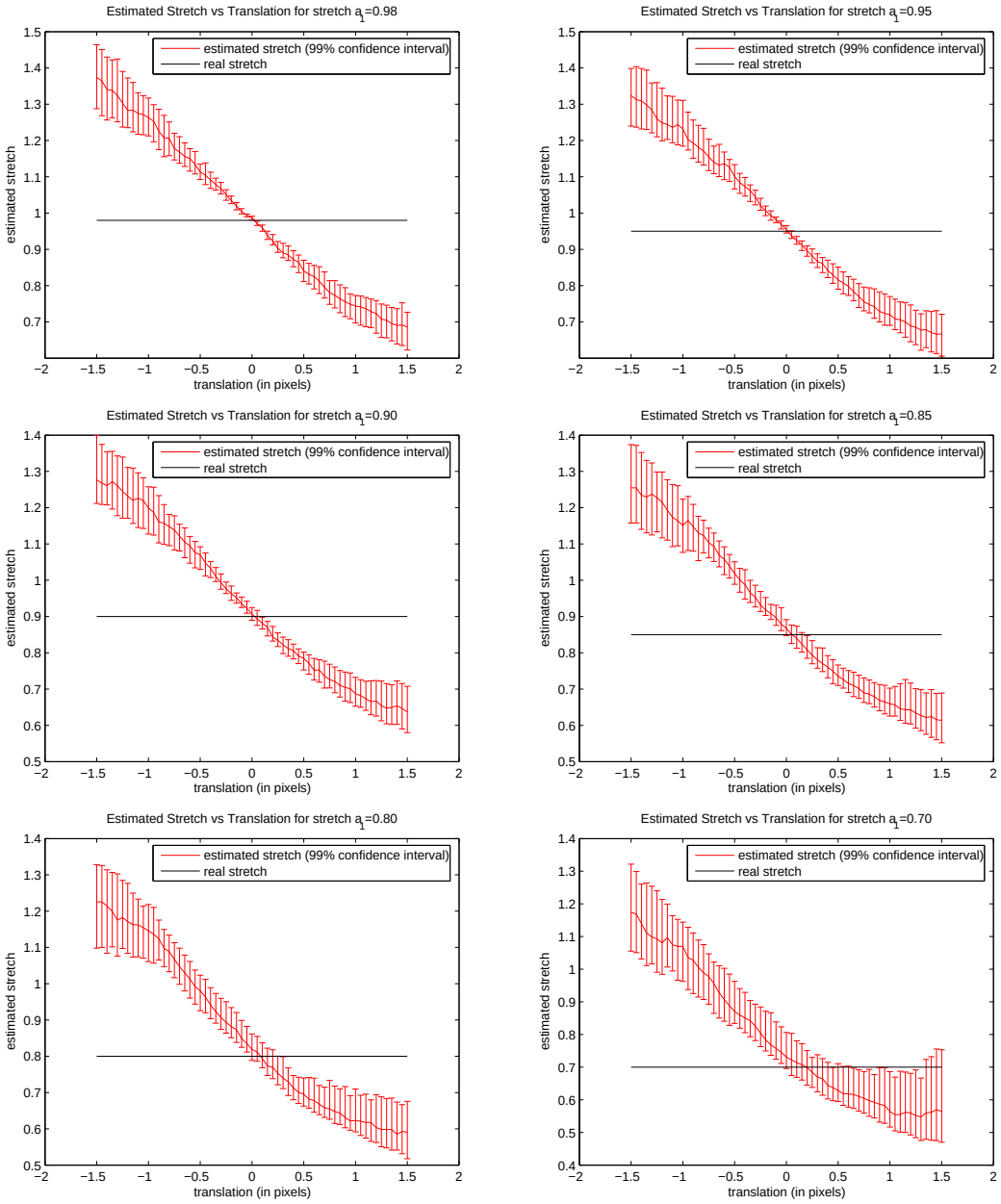


Figure B.6: Stretch estimation results with a single filter when there is a shift error. Each figure corresponds to a different actual stretch value.The error curve produced when the two signals are not perfectly aligned does not depend much on the actual stretch value.

estimates, when the two signals are stretched and shifted, as a function of the shift. As expected (due to the small size of the filters), this approach is very sensitive to shifts. Furthermore, the error in the stretch estimation increases with the shift.

B.5 Conclusions

In this chapter we presented a filter that retrieves the local stretch of two signals. We also presented experiments that indicate that this approach produces very good results, but is also very sensitive to the shift between the two signals. Two simple improvements that will decrease the sensitivity to the shift error and increase the accuracy of the estimation are

- use the stretch results from multiple filters. We noticed that filters with different u values exhibit different sensitivity to shift errors and can work accurately for different ranges of stretches. The smaller the value of u the more sensitive the filter is to shift noise, but at the same time the more accurate the results are for a larger range of stretches. Thus a carefully constructed band of stretch filters could provide high noise tolerance and high stretch sensitivity.

- compute the stretch from more signal points. It is experimentally established that frequency based approaches for registering two signals require as much data points as possible. In the previous experiments the filter size was around 20 pixels. A larger filter would be much less sensitive to shifts. Furthermore since we work with images considering multiple scan lines in the computation of stretch would further improve the robustness and accuracy of the results.

If both improvements are carefully implemented then this method can potentially provide a real-time alternative for stretch estimation and could be used in real systems.

Appendix C

Towards Surface Segmentation Proofs

Lemma C.1. *If we model the error in the localization of the feature points as independent Gaussian random variables with variance σ^2, σ'^2 for the features on the first and second frame respectively, the 9×9 covariance matrix of the homography is*

$$C_{\mathbf{h}} = JSJ^T \tag{C.1}$$

,where

$$J = [\ \mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_9\ ] \begin{bmatrix} 0 & 0 & \dots & 0 & \mathbf{x}_1^T \\ 0 & \frac{1}{\lambda_1-\lambda_2} & \dots & 0 & \mathbf{x}_2^T \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda_1-\lambda_9} & \mathbf{x}_9^T \end{bmatrix} \tag{C.2}$$

,with $\mathbf{x}_i$ the eigenvector corresponding to the i^{th} eigenvalue λ_i of matrix A^TA. Matrix S is

$$S = \sum_{i=1}^{n} (\mathbf{r}_{2i}^T \mathbf{r}_{2i} f_i^e + \mathbf{r}_{2i-1}^T \mathbf{r}_{2i-1} f_i^o + \mathbf{r}_{2i}^T \mathbf{r}_{2i-1} f_i^{eo} + \mathbf{r}_{2i-1}^T \mathbf{r}_{2i} f_i^{oe}) \tag{C.3}$$

with $\mathbf{r}_i$ *the* i^{th} *row of matrix* A *and*

$$\begin{aligned}
f_i^e = \;& \sigma^2[h_1^2 + h_2^2 - 2x_i'(h_1h_7 + h_2h_8)] + 2\sigma'^2(x_ih_7h_9 + x_iy_ih_7h_8 + y_ih_8h_9) \\
& + (\sigma^2 x_i'^2 + x_i^2\sigma'^2)h_7^2 + (\sigma^2 x_i'^2 + y_i^2\sigma'^2)h_8^2 + \sigma'^2h_9^2 \\
f_i^o = \;& \sigma^2[h_4^2 + h_5^2 - 2y_i'(h_4h_7 + h_5h_8)] + 2\sigma'^2(x_ih_7h_9 + x_iy_ih_7h_8 + y_ih_8h_9) \\
& + (\sigma^2 y_i'^2 + x_i^2\sigma'^2)h_7^2 + (\sigma^2 y_i'^2 + y_i^2\sigma'^2)h_8^2 + \sigma'^2h_9^2 \\
f_i^{oe} = f_i^{eo} = \;& \sigma^2[(h_1 - x_i'h_7)(h_4 - y_i'h_7) + (h_2 - x_i'h_8)(h_5 - y_i'h_8)]
\end{aligned}$$

Proof. This proof is based on eigenvector perturbation theory and is similar to proofs given in [95, 120]. Next we provide a detailed outline of the proof, but omit the final cumbersome algebraic computations.

First a note on notation. We use the subscript zero $:_0$ to denote the initial measurement and the symbol δ for the perturbation matrix or vector.

Let us denote with B_0 the 9×9 matrix of the product of the original measurements $A_0^TA_0$. The perturbed matrix $A = A_0 + \delta A$ introduces a perturbed matrix $B = A^TA = (A_0 + \delta A)^T(A_0 + \delta A) = A_0^TA_0 + A_0^T\delta A + \delta A^TA_0 + \delta A^T\delta A \simeq A_0^TA_0 + A_0^T\delta A + \delta A^TA_0$. As you notice we only keep the linear error terms and drop the higher order error terms. Hence

$$B = B_0 + \delta B \tag{C.4}$$

, where

$$\delta B = \delta A^TA + A^T\delta A. \tag{C.5}$$

Let us denote with $\lambda_i,\ \mathbf{x}_i$ the i^{th} eigenvalue and eigenvector respectively of matrix B $(i = 1 \ldots 9)$. Our goal is to find an analytic expression for the i^{th} eigenvector $\mathbf{x}_i$ and eigenvalue λ_i with respect to the perturbation matrix δB and the eigenvectors and eigenvalues of the initial matrix B_0 $(\mathbf{x}_{0i},\ \lambda_{0i})$. If we express the new measurements as

$$\lambda_i = \lambda_{0i} + \delta\lambda_i, \tag{C.6}$$

$$\mathbf{x}_i = \mathbf{x}_{0i} + \delta\mathbf{x}_i \tag{C.7}$$

one needs to compute the differences $\delta\lambda_i,\ \delta\mathbf{x}_i$.

From the definition of the eigenvalues and eigenvectors we have

$$B_0\mathbf{x}_{0i} = \lambda_{0i}\mathbf{x}_{0i} \tag{C.8}$$

The same equation is valid for the new eigenvalues and eigenvectors i.e.,

$$B\mathbf{x}_i = \lambda_i\mathbf{x}_i \Rightarrow \tag{C.9}$$

$$(B_0 + \delta B)(\mathbf{x}_{0i} + \delta\mathbf{x}_i) = (\lambda_{0i} + \delta\lambda_i)(\mathbf{x}_{0i} + \delta\mathbf{x}_i) \Rightarrow \tag{C.10}$$

$$B_0\mathbf{x}_{0i} + B_0\delta\mathbf{x}_i + \delta B\mathbf{x}_{0i} + \delta B\delta\mathbf{x}_i = \lambda_{0i}\mathbf{x}_{0i} + \lambda_{0i}\delta\mathbf{x}_i + \delta\lambda_i\mathbf{x}_{0i} + \delta\lambda_i\delta\mathbf{x}_i. \tag{C.11}$$

From Eq. C.8 and ignoring the second order terms we simplify Eq. C.11 as

$$B_0\delta\mathbf{x}_i + \delta B\mathbf{x}_{0i} = \lambda_{0i}\delta\mathbf{x}_i + \delta\lambda_i\mathbf{x}_{0i}. \tag{C.12}$$

The eigenvectors of the original matrix B_0 form a coordinate system for the 9D space so we can express the eigenvector change $\delta\mathbf{x}_i$ as a linear combination of $\mathbf{x}_{0i}$ i.e.,

$$\delta\mathbf{x}_i = \sum_{j=1}^{9} \epsilon_{ij}\mathbf{x}_{0j}. \tag{C.13}$$

Combining Eqs. C.12, C.13 we obtain

$$B_0\sum_{j=1}^{9} \epsilon_{ij}\mathbf{x}_{0j} + \delta B\mathbf{x}_{0i} = \lambda_{0i}\sum_{j=1}^{9} \epsilon_{ij}\mathbf{x}_{0j} + \delta\lambda_i\mathbf{x}_{0i}, \tag{C.14}$$

$$\sum_{j=1}^{9} \epsilon_{ij}\lambda_{0j}\mathbf{x}_{0j} + \delta B\mathbf{x}_{0i} = \lambda_{0i}\sum_{j=1}^{9} \epsilon_{ij}\mathbf{x}_{0j} + \delta\lambda_i\mathbf{x}_{0i}. \tag{C.15}$$

Left multiply Eq. C.15 with $\mathbf{x}_{0i}^T$ and considering the orthogonality of $\mathbf{x}_{0i}$ i.e.,

$$\mathbf{x}_{0i}^T\mathbf{x}_{0j} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{C.16}$$

we get

$$\epsilon_{ii}\lambda_{0i}\mathbf{x}_{0i}^T\mathbf{x}_{0i} + \mathbf{x}_{0i}^T\delta B\mathbf{x}_{0i} = \lambda_{0i}\epsilon_{ii}\mathbf{x}_{0i}^T\mathbf{x}_{0i} + \delta\lambda_i\mathbf{x}_{0i}^T\mathbf{x}_{0i} \Rightarrow \tag{C.17}$$

$$\delta\lambda_i = \mathbf{x}_{0i}^T \delta B \mathbf{x}_{0i}. \tag{C.18}$$

Left multiply Eq. C.15 with x_{0k}^T, $k \neq i$ we get

$$\mathbf{x}_{0k}^T \sum_{j=1}^{9} \epsilon_{ij}\lambda_{0j}\mathbf{x}_{0j} + \mathbf{x}_{0k}^T \delta B \mathbf{x}_{0i} = \lambda_{0i}\mathbf{x}_{0k}^T \sum_{j=1}^{9} \epsilon_{ij}\mathbf{x}_{0j} + \mathbf{x}_{0k}^T \delta\lambda_i \mathbf{x}_{0i} \Rightarrow \tag{C.19}$$

$$\mathbf{x}_{0k}^T \epsilon_{ik}\lambda_{0k}\mathbf{x}_{0k} + \mathbf{x}_{0k}^T \delta B \mathbf{x}_{0i} = \lambda_{0i}\epsilon_{ik}\mathbf{x}_{0k}^T \mathbf{x}_{0k} \Rightarrow \tag{C.20}$$

$$\epsilon_{ik}\lambda_{0k} + \mathbf{x}_{0k}^T \delta B \mathbf{x}_{0i} = \lambda_{0i}\epsilon_{ik} \Rightarrow \tag{C.21}$$

$$\epsilon_{ik} = \frac{\mathbf{x}_{0k}^T \delta B \mathbf{x}_{0i}}{\lambda_{0i} - \lambda_{0k}}, \quad k \neq i. \tag{C.22}$$

In order to compute the remaining coefficients ϵ_{ii} we use the orthogonality of the eigenvectors i.e.,

$$\mathbf{x}_i^T \mathbf{x}_i = 1 \Rightarrow \tag{C.23}$$

$$(\mathbf{x}_{0i} + \delta\mathbf{x}_i)^T (\mathbf{x}_{0i} + \delta\mathbf{x}_i) = 1 \Rightarrow \tag{C.24}$$

$$\mathbf{x}_{0i}^T\mathbf{x}_{0i} + \mathbf{x}_{0i}^T\delta\mathbf{x}_i + \delta\mathbf{x}_i^T\mathbf{x}_{0i} + \delta\mathbf{x}_i^T\delta\mathbf{x}_i = 1 \tag{C.25}$$

Ignoring the second order term $\delta\mathbf{x}_i^T\delta\mathbf{x}_i$ and given the fact that $\mathbf{x}_{0i}^T\mathbf{x}_{0i} = 1$ we get

$$\mathbf{x}_{0i}^T\delta\mathbf{x}_i + \delta\mathbf{x}_i^T\mathbf{x}_{0i} = 0 \tag{C.26}$$

From Eq. C.13 and left multiplying with $\mathbf{x}_{0i}^T$ we get

$$\mathbf{x}_{0i}^T\sum_{j=1}^{9}\epsilon_{ij}\mathbf{x}_{0j} + (\sum_{j=1}^{9}\epsilon_{ij}\mathbf{x}_{0j})^T\mathbf{x}_{0i} = 0 \Rightarrow \tag{C.27}$$

$$\epsilon_{ii} = 0. \tag{C.28}$$

In synopsis a perturbed matrix B by δB causes a change in the eigenvalue λ_i by

$$\delta\lambda_i = \mathbf{x}_{0i}^T\delta B\mathbf{x}_{0i} \tag{C.29}$$

and a corresponding change in the eigenvector $\mathbf{x}_i$ by

$$\delta\mathbf{x}_i = \sum_{j=1,j\neq i}^{9}(\frac{\mathbf{x}_{0j}^T\delta B\mathbf{x}_{0i}}{\lambda_{0i} - \lambda_{0j}})\mathbf{x}_{0j}. \tag{C.30}$$

Since the homography is the eigenvector corresponding to the least eigenvalue (i.e.,$\mathbf{h} \equiv \mathbf{x}_{01}$), if we rewrite the sum of Eq. C.30 in matrix form we get

$$\delta\mathbf{h} = \begin{bmatrix} \mathbf{x}_{01} & \mathbf{x}_{02} & \dots & \mathbf{x}_{09} \end{bmatrix} \begin{bmatrix} 0 & 0 & \dots & 0 & \mathbf{x}_{01}^T \\ 0 & \frac{1}{\lambda_{01}-\lambda_{02}} & \dots & 0 & \mathbf{x}_{02}^T \\ \dots & \dots & \ddots & \dots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda_{01}-\lambda_{09}} & \mathbf{x}_{09}^T \end{bmatrix} \delta B\mathbf{h} \tag{C.31}$$

We want to express the change in the homography as a linear combination of the elements of matrix δB. If we denote the identity matrix of size 9×9 as I_9 and the individual elements of vector $\mathbf{h}$ as $h_1,\ h_2, \dots, h_9$ then

$$\delta B\mathbf{h} \equiv \begin{bmatrix} h_1 I_9 & h_2 I_9 & \dots & h_9 I_9 \end{bmatrix} \delta\mathbf{b} \tag{C.32}$$

, where $\delta\mathbf{b}$ is a column vector produced by δB as follows

$$\delta b = \begin{bmatrix} \delta B_{11} & \delta B_{21} & \dots & \delta B_{91} & \delta B_{12} & \dots & \delta B_{92} & \dots & \delta B_{91} & \dots & \delta B_{99} \end{bmatrix}^T. \tag{C.33}$$

Next we need to express the perturbation vector $\delta\mathbf{b}$ with respect to the perturbation vector $\delta\mathbf{a}$. As above we get $\delta\mathbf{a}$ by concatenating the columns of matrix δA^T. From Eq. C.5 by denoting with $\delta b_i,\ \delta B_{ij},\ \delta a_{ij},\ a_{ij}$ the elements of the i^{th} row and j^{th} column of matrices $\delta\mathbf{b},\ \delta B,\ \delta A,\ A$ respectively with a little algebra we obtain the following expression

$$\delta b_{9(j-1)+i} = \delta B_{ij} = \sum_{k=1}^{2n} (a_{ki}\delta a_{kj} + \delta a_{ki} a_{kj}) \tag{C.34}$$

that is linear on the perturbation vector $\delta\mathbf{a}$ i.e.,

$$\delta\mathbf{b} = G\delta\mathbf{a} \tag{C.35}$$

for a properly constructed $81 \times (2n \cdot 9)$ matrix G.

The last step is to compute the covariance matrix for matrix A. The homography estimation matrix A is

$$A = \begin{bmatrix} 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -x_n & -y_n & -1 & y'_nx_n & y'_ny_n & y'_n \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_nx_n & -x'_ny_n & -x'_n \end{bmatrix}. \tag{C.36}$$

If we assume that the components of the image vectors $\mathbf{x}_i = (x_i,\ y_i,\ 1)^T$, $\mathbf{x}'_i = (x'_i,\ y'_i,\ 1)^T$ have errors $(\delta x_i,\ \delta y_i,\ \delta x'_i,\ \delta y'_i)$ then we get the perturbation matrix

$$\delta A^T = \begin{bmatrix} 0 & 0 & 0 & \delta x_1 & \delta y_1 & 0 & -(x_1\delta y'_1 + y'_1\delta x_1) & -(y_1\delta y'_1 + y'_1\delta y_1) & -\delta y'_1 \\ \delta x_1 & \delta y_1 & 0 & 0 & 0 & 0 & -(x_1\delta x'_1 + x'_1\delta x_1) & -(y_1\delta x'_1 + x'_1\delta y_1) & -\delta x'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \delta x_n & \delta y_n & 0 & -(x_n\delta y'_n + y'_n\delta x_n) & -(y_n\delta y'_n + y'_n\delta y_n) & -\delta y'_n \\ \delta x_n & \delta y_n & 0 & 0 & 0 & 0 & -(x_n\delta x'_n + x'_n\delta x_n) & -(y_n\delta x'_n + x'_n\delta y_n) & -\delta x'_n \end{bmatrix}^T. \tag{C.37}$$

If we model these errors as independent, random variables following Gaussian distributions with zero mean value and variance σ^2, σ'^2 for the first and second image respectively we can compute the covariance matrix of δA^T. As we mentioned above we create the vector $\delta\mathbf{a}$ by concatenating the columns of δA^T. Vector $\delta\mathbf{a}$ has $18n$ entries, thus the covariance matrix of δA^T has a size of $18n \times 18n$. Since the variables are independent, the covariance matrix $C_{\mathbf{a}}$ has a block diagonal form $C_{\mathbf{a}} = diag\{E_1,\ E_2, \ldots,\ E_n\}$ with the 18×18 diagonal elements being displayed on Fig. C.1.

From Eqs. C.31, C.32 and C.35 we get that the perturbation of the homography vector $\delta\mathbf{h}$ is a linear combination of the perturbation of the input vector $\delta\mathbf{a}$ i.e.,

$$\delta\mathbf{h} = \begin{bmatrix} \mathbf{x}_{01} & \mathbf{x}_{02} & \ldots & \mathbf{x}_{09} \end{bmatrix} \begin{bmatrix} 0 & 0 & \ldots & 0 & \mathbf{x}_{01}^T \\ 0 & \frac{1}{\lambda_{01}-\lambda_{02}} & \ldots & 0 & \mathbf{x}_{02}^T \\ \ldots & \ldots & \ddots & \ldots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\lambda_{01}-\lambda_{09}} & \mathbf{x}_{09}^T \end{bmatrix} \begin{bmatrix} h_1 I_9 & h_2 I_9 & \cdots & h_9 I_9 \end{bmatrix} G\delta\mathbf{a}. \tag{C.39}$$

If we compact the notation by using the matrix J

$$J = \begin{bmatrix} \mathbf{x}_{01} & \mathbf{x}_{02} & \ldots & \mathbf{x}_{09} \end{bmatrix} \begin{bmatrix} 0 & 0 & \ldots & 0 & \mathbf{x}_{01}^T \\ 0 & \frac{1}{\lambda_{01}-\lambda_{02}} & \ldots & 0 & \mathbf{x}_{02}^T \\ \ldots & \ldots & \ddots & \ldots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\lambda_{01}-\lambda_{09}} & \mathbf{x}_{09}^T \end{bmatrix}, \tag{C.40}$$

$$E_i = \left[\begin{array}{cccccccccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma^2 & 0 & 0 & -y_i'\sigma^2 & 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 \\
0 & 0 & 0 & 0 & \sigma^2 & 0 & 0 & -y_i'\sigma^2 & 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & -x_i'\sigma^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -y_i'\sigma^2 & 0 & 0 & x_i^2\sigma'^2 + y_i'^2\sigma^2 & x_i y_i \sigma'^2 & x_i\sigma'^2 & -y_i'\sigma^2 & 0 & 0 & 0 & 0 & 0 & x_i' y_i' \sigma^2 & 0 & 0 \\
0 & 0 & 0 & 0 & -y_i'\sigma^2 & 0 & x_i y_i \sigma'^2 & y_i^2\sigma'^2 + y_i'^2\sigma^2 & y_i\sigma'^2 & 0 & -y_i'\sigma^2 & 0 & 0 & 0 & 0 & 0 & x_i' y_i' \sigma^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x_i\sigma'^2 & y_i\sigma'^2 & \sigma'^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \sigma^2 & 0 & 0 & -y_i'\sigma^2 & 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 \\
0 & 0 & 0 & 0 & \sigma^2 & 0 & 0 & -y_i'\sigma^2 & 0 & 0 & \sigma^2 & 0 & 0 & 0 & 0 & 0 & -x_i'\sigma^2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 & x_i' y_i' \sigma^2 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 & 0 & 0 & 0 & x_i^2\sigma'^2 + x_i'^2\sigma^2 & x_i y_i \sigma'^2 & x_i\sigma'^2 \\
0 & 0 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 & x_i' y_i' \sigma^2 & 0 & 0 & -x_i'\sigma^2 & 0 & 0 & 0 & 0 & x_i y_i \sigma'^2 & y_i^2\sigma'^2 + x_i'^2\sigma^2 & y_i\sigma'^2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_i\sigma'^2 & y_i\sigma'^2 & \sigma'^2
\end{array} \right] \tag{C.38}$$

Figure C.1: Block i of the covariance matrix for $\delta\mathbf{a}$

the covariance matrix $C_{\mathbf{h}}$ is given by

$$C_{\mathbf{h}} = J[\ h_1 I_9 \quad h_2 I_9 \quad \cdots \quad h_9 I_9\ ] G C_{\mathbf{a}} G^T [\ h_1 I_9 \quad h_2 I_9 \quad \cdots \quad h_9 I_9\ ]^T J^T \qquad \text{(C.41)}$$

With the proper algebraic manipulation we get the final result. □

Lemma C.2. *Given the projection matrices for the two views of a camera with intrinsic parameters* $\mathbf{K}$

$$\mathbf{P} = \mathbf{K} \cdot [\mathbf{I} \mid \mathbf{0}] \qquad \mathbf{P}' = \mathbf{K} \cdot [\mathbf{R} \mid \mathbf{t}] \tag{C.42}$$

where $\mathbf{R}$, $\mathbf{t}$ *represent the rotation and translation between the two views respectively and a plane defined by* $\pi^T \cdot \mathbf{X} = 0$ *with* $\pi = (\nu^T,\ 1)^T$ *(ν is the surface normal), then the homography induced by the plane is* $\mathbf{x}' = \mathbf{H} \cdot \mathbf{x}$ *with*

$$\mathbf{H} = \mathbf{K} \cdot (\mathbf{R} - \mathbf{t} \cdot \nu^T) \cdot \mathbf{K}^{-1}. \tag{C.43}$$

Proof. The general idea is to compute the world point with respect to the image point of the first and second frame and equate the two expressions. Let us assume that there is a plane π with surface normal $\mathbf{N} = (\nu^T, 1)^T$ in homogeneous coordinates[1]. By definition any world point $\mathbf{X}$ belonging to that plane satisfies the equation

$$\mathbf{N^T X} = 0. \tag{C.44}$$

Let $\mathbf{x}$, $\mathbf{x}'$ denote the projection of the world point on the first and second frame respectively. If $\mathbf{P} = \mathbf{K} \cdot [\mathbf{I} \mid \mathbf{0}]$ and $\mathbf{P}' = \mathbf{K} \cdot [\mathbf{R} \mid \mathbf{t}]$ are the projection matrices for the two camera views then $\mathbf{x} = \mathbf{P} \cdot \mathbf{X}$ and $\mathbf{x}' = \mathbf{P}' \cdot \mathbf{X}$ respectively. If we parameterize the world point $\mathbf{X} = (\mathbf{y^T},\ \rho)^T$ then we get that $\mathbf{x} = \mathbf{K} \cdot [\mathbf{I} \mid \mathbf{0}] \cdot (\mathbf{y^T},\ \rho)^T = \mathbf{K} \cdot \mathbf{y}$. Thus the world point $\mathbf{X}$ belongs to the ray parameterized by $\rho\, \mathbf{X} = ((\mathbf{K}^{-1} \cdot \mathbf{x})^T,\ \rho)^T$.

[1] We assume that the plane does not pass through the center of the camera of the first frame at $(0,\ 0,\ 0,\ 1)^T$ that's why we are allowed to assume that $\pi_4 = 1$.

Using Eq. C.44 we compute $\rho = -\nu^T \mathbf{K}^{-1}\mathbf{x}$, thus

$$\mathbf{X} = ((\mathbf{K}^{-1}\mathbf{x})^T,\ \nu^T \mathbf{K}^{-1}\mathbf{x})^T \tag{C.45}$$

From the projection of $\mathbf{X}$ to the second view we get

$$\mathbf{x}' = \mathbf{K} \cdot [\mathbf{R} \mid \mathbf{t}] \cdot ((\mathbf{K}^{-1}\mathbf{x})^T,\ \nu^T \mathbf{K}^{-1}\mathbf{x})^T = \mathbf{K}(\mathbf{R} - \mathbf{t}\nu^T)\mathbf{K}^{-1}\mathbf{x} \tag{C.46}$$

□

Appendix D

PTU-Camera calibration

In this chapter we describe the procedure of calibrating the Pan and Tilt Unit (PTU) with the cameras. Fig. **D.1** displays the setting that we use in our experiments. An array of cameras is attached on top of a Pan and Tilt Unit (PTU). The PTU has two degrees of freedom namely rotation around the horizontal (pan) and vertical (tilt) plane. In the experiments we use a single camera (located on the top right corner of the array). Before performing any experiment we have to calibrate the camera with respect to the PTU. The next section describes that procedure in details.

D.1 Acquiring calibration data

We captured images of a checkerboard pattern for different pan and tilt angles. We selected 11 different pan and tilt angles and calibrated for the pan and the tilt independently. We captured 10 images for each angle for a total of 220 images. Figs. D.2, D.3 display some of these images.

We used the camera calibration toolbox created by Bouguet [121] to compute the intrinsic parameters first. Then, selecting the 0° pan, 0° tilt set of images as baseline, we computed the extrinsic parameters for each different pan/tilt combination with respect to the baseline. Table D.1 displays the rotation and translation of

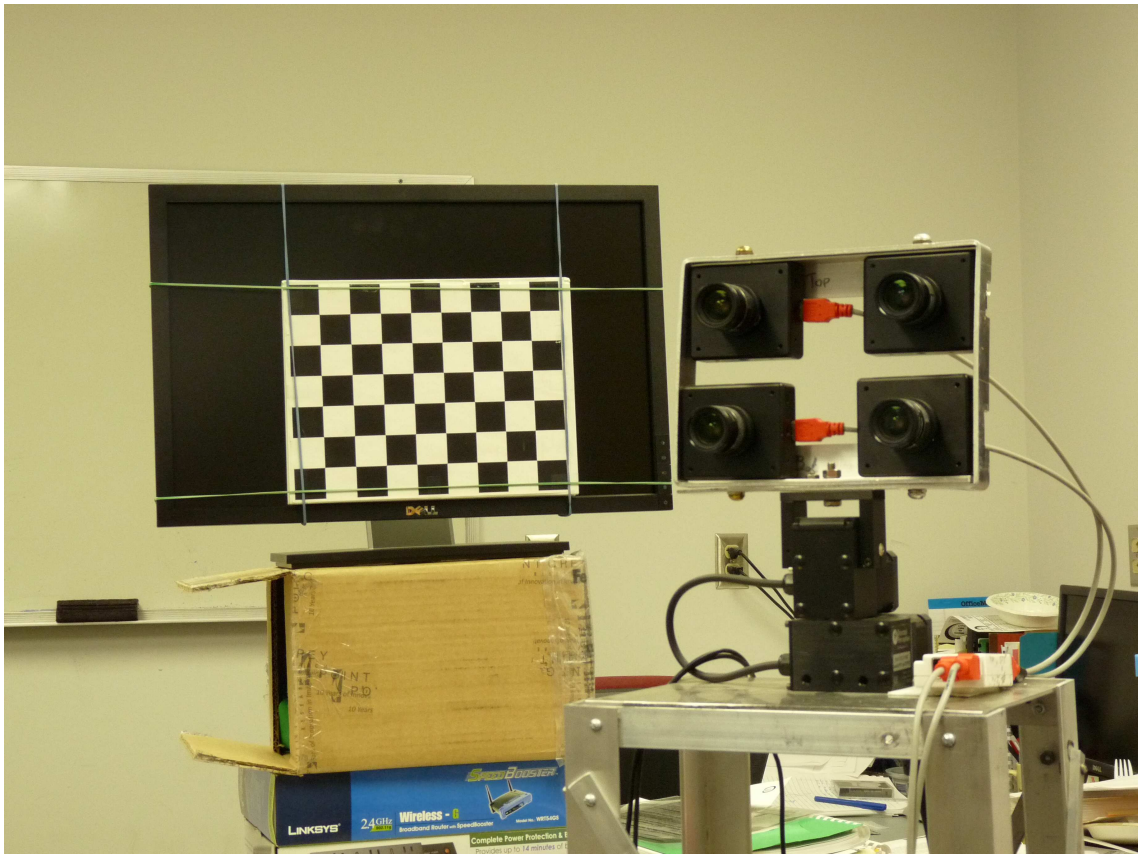


Figure D.1: The Pan and Tilt Unit (PTU) and the cameras attached to it.

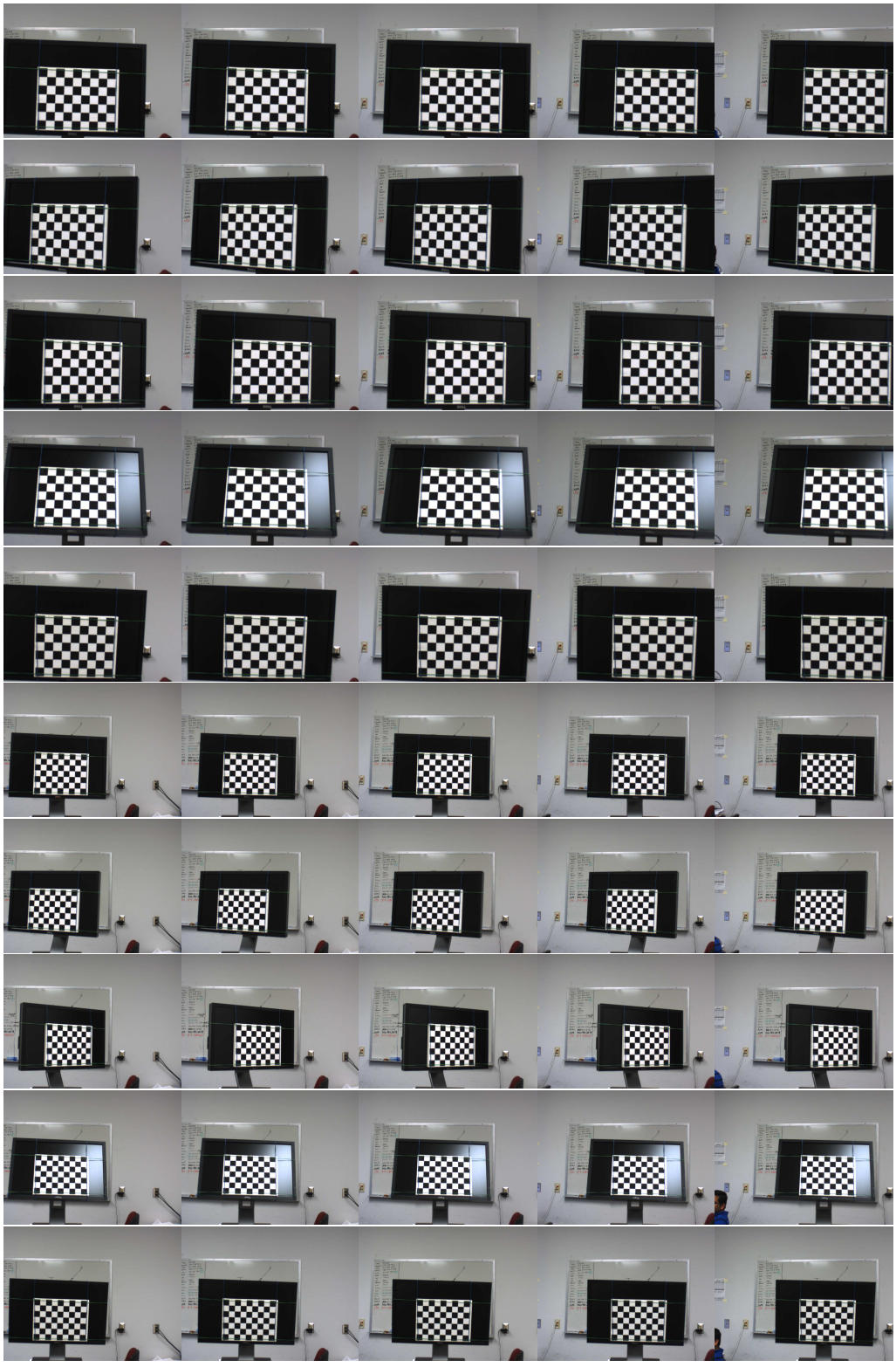


Figure D.2: Calibration images for different pan values. Each column represents a different pan angle $(-5^{\circ}, -3^{\circ}, 0^{\circ}, 3^{\circ}, 5^{\circ})$ and each row a different placement of the calibration grid.

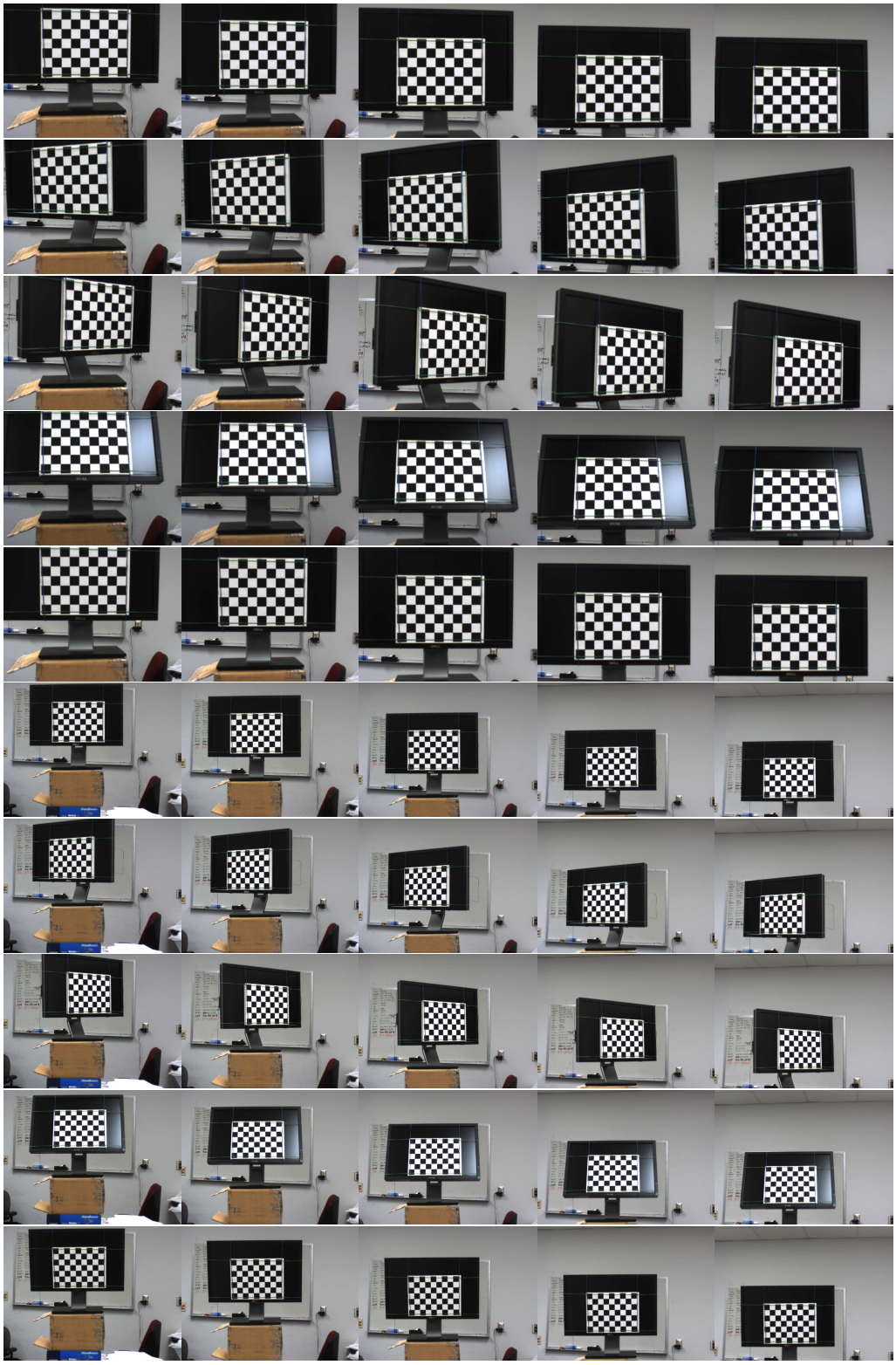
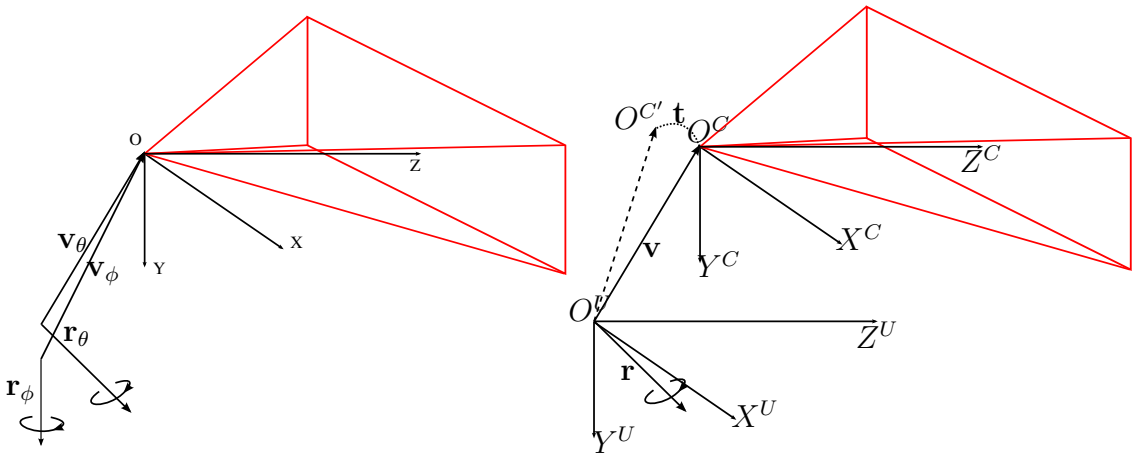


Figure D.3: Calibration images for different tilt values. Each column represents a different tilt angle $(-5^\circ, -3^\circ, 0^\circ, 3^\circ, 5^\circ)$ and each row a different placement of the calibration grid.



(a) The camera coordinate system and the two rotation axis for the PTU

(b) The camera and PTU coordinate systems

Figure D.4: The camera and PTU coordinate systems.

the camera that corresponds to pan and tilt rotations of the PTU[1]. More specifically the translation vector displays the new position of the camera center with respect to the coordinate system of the camera at 0° pan and 0° tilt.

In Fig. D.4(a) we draw the camera coordinate system that we use in the rest of this section.

D.2 Calibrating the camera with respect to the PTU

Our goal is to analytically compute an estimate for the rotation and translation of the camera as we rotate and translate the PTU (Fig. D.4(a)). More specifically we need to estimate

1. the two axes of rotation $(\mathbf{r}_\theta, \mathbf{r}_\phi)$ and the corresponding vectors $(\mathbf{v}_\theta, \mathbf{v}_\phi)$ between the center of the two rotation axes and the focal point of the camera that cause the translation of the camera center

[1]The numerical errors are approximately three times the standard deviations

Table D.1: Rotation and translation of the camera with respect to the baseline position (0° tilt, 0° pan).

Pan (ϕ)	Rotation Vector (ω_ϕ)			Rotation Angle	Translation Vector (t)		
-5°	-0.022 ± 0.086	-0.998 ± 0.131	0.053 ± 0.006	$5.07^\circ \pm 0.69^\circ$	-1.349 ± 0.198	0.215 ± 0.134	5.422 ± 1.460
-4°	-0.014 ± 0.093	-0.998 ± 0.142	0.053 ± 0.006	$4.32^\circ \pm 0.64^\circ$	-0.442 ± 0.160	0.057 ± 0.119	2.975 ± 1.348
-3°	-0.004 ± 0.109	-0.999 ± 0.166	0.053 ± 0.007	$3.32^\circ \pm 0.57^\circ$	-0.358 ± 0.126	0.087 ± 0.105	2.173 ± 1.211
-2°	-0.046 ± 0.167	-0.998 ± 0.255	0.053 ± 0.010	$2.12^\circ \pm 0.58^\circ$	-0.490 ± 0.107	0.082 ± 0.104	1.771 ± 1.192
-1°	-0.086 ± 0.348	-0.995 ± 0.535	0.054 ± 0.020	$1.03^\circ \pm 0.61^\circ$	-0.259 ± 0.092	0.055 ± 0.102	1.057 ± 1.184
1°	0.125 ± 0.320	0.991 ± 0.487	-0.049 ± 0.018	$1.18^\circ \pm 0.64^\circ$	0.321 ± 0.080	-0.045 ± 0.104	-0.953 ± 1.232
2°	0.027 ± 0.194	0.998 ± 0.295	-0.052 ± 0.011	$2.05^\circ \pm 0.64^\circ$	0.567 ± 0.086	-0.099 ± 0.112	-2.783 ± 1.322
3°	0.022 ± 0.153	0.998 ± 0.233	-0.053 ± 0.009	$2.95^\circ \pm 0.73^\circ$	0.956 ± 0.111	-0.226 ± 0.131	-5.010 ± 1.522
4°	0.002 ± 0.117	0.999 ± 0.181	-0.053 ± 0.007	$4.02^\circ \pm 0.75^\circ$	1.176 ± 0.135	-0.239 ± 0.140	-6.073 ± 1.602
5°	0.029 ± 0.091	0.998 ± 0.142	-0.053 ± 0.006	$5.12^\circ \pm 0.76^\circ$	1.287 ± 0.157	-0.157 ± 0.140	-7.203 ± 1.594

Tilt (θ)	Rotation Vector (ω_θ)			Rotation Angle	Translation Vector (t)		
-5°	0.997 ± 0.190	-0.078 ± 0.239	0.014 ± 0.012	$4.86^\circ \pm 1.11^\circ$	0.946 ± 0.265	-1.258 ± 0.328	-12.655 ± 2.252
-4°	0.996 ± 0.177	-0.090 ± 0.220	0.015 ± 0.010	$3.89^\circ \pm 0.83^\circ$	0.667 ± 0.178	-1.236 ± 0.213	-10.310 ± 1.684
-3°	0.995 ± 0.186	-0.102 ± 0.236	0.015 ± 0.009	$2.92^\circ \pm 0.67^\circ$	0.679 ± 0.127	-0.739 ± 0.142	-7.654 ± 1.330
-2°	0.981 ± 0.229	-0.193 ± 0.290	0.009 ± 0.010	$2.09^\circ \pm 0.63^\circ$	0.379 ± 0.104	-0.579 ± 0.104	-5.296 ± 1.180
-1°	0.998 ± 0.452	-0.058 ± 0.562	0.018 ± 0.018	$0.98^\circ \pm 0.57^\circ$	0.155 ± 0.088	-0.331 ± 0.084	-2.622 ± 1.098
1°	-0.991 ± 0.499	0.135 ± 0.623	-0.017 ± 0.017	$0.90^\circ \pm 0.60^\circ$	-0.190 ± 0.094	0.428 ± 0.077	2.835 ± 1.123
2°	-0.987 ± 0.239	0.160 ± 0.315	-0.012 ± 0.008	$2.05^\circ \pm 0.64^\circ$	-0.361 ± 0.116	0.875 ± 0.093	5.323 ± 1.229
3°	-0.999 ± 0.158	-0.035 ± 0.220	-0.019 ± 0.006	$3.06^\circ \pm 0.57^\circ$	-0.372 ± 0.140	1.296 ± 0.110	8.403 ± 1.221
4°	-0.999 ± 0.123	-0.051 ± 0.177	-0.018 ± 0.005	$3.99^\circ \pm 0.58^\circ$	-0.513 ± 0.169	1.847 ± 0.136	11.477 ± 1.240
5°	-0.996 ± 0.096	-0.091 ± 0.139	-0.020 ± 0.005	$5.01^\circ \pm 0.58^\circ$	-0.690 ± 0.187	2.491 ± 0.157	14.089 ± 1.210

2. the two axes of rotation $(\omega_\theta, \omega_\phi)$ that rotate the coordinate system attached to the camera center.

In the following subsections we describe both procedures. Note that we use the symbols $\omega_\phi, \omega_\theta$ for the rotation of the whole camera coordinate system, while the symbols r_ϕ, r_θ for the rotation axes of the PTU unit. Also we denote with ϕ, θ the pan and tilt angles of the PTU, while with ψ, ξ the angles we use for the parametrization of the rotation axes r_ϕ, r_θ.

D.2.1 Estimating the translation of the camera center

Let us denote with $O^C X^C Y^C Z^C$ the coordinate system attached to the camera at the 0° pan and 0° tilt position. The translation measurements of Table D.1 correspond to the position of the camera center when we pan or tilt the camera at a given angle. We denote that position with $O^{C'}$ in Fig. **D.4**(b). We consider the coordinate system of the PTU $O^U X^U Y^U Z^U$ to be a translated version of $O^C X^C Y^C Z^C$ by v. Hence, for any point $\mathbf{P}$ the relation of its coordinates in the two coordinate systems is $\mathbf{P}^U = \mathbf{P}^C + \mathbf{v}$. Then, we apply the rotation around axis $\mathbf{r}$, so the camera center moves from O_C to $O_{C'}$. Denoting the rotation matrix (corresponding to the rotation axis $\mathbf{r}$) with $\mathbf{R}(\mathbf{r})$ we have the following equations

$$\begin{aligned} O_{C'} &= \mathbf{R}(\mathbf{r}) \cdot \mathbf{v} \\ O_{C'} &= \mathbf{v} + \mathbf{t}. \end{aligned}$$

The above equations leads us to the bilinear system with respect to the rotation matrix $\mathbf{R}$ and the vector $\mathbf{v}$

$$(\mathbf{R}(\mathbf{r}) - \mathbf{I})\mathbf{v} = \mathbf{t}. \tag{D.1}$$

Using the translation of the camera center ($\mathbf{t}_i$) for different angles (θ_i) of Table D.1, we need to estimate both the rotation axis $\mathbf{r}$ (2 parameters) and the vector $\mathbf{v}$ (3 parameters). Using the Rodrigues formula to express the rotation around an axis

$$\mathbf{R} = (1 - \cos\theta)\mathbf{r} \cdot \mathbf{r}^T + \cos\theta\mathbf{I} + \sin\theta\mathbf{Q}_\mathbf{r} \tag{D.2}$$

, where $\mathbf{Q}_\mathbf{r} = \begin{bmatrix} 0 & -r_3 & r_2 \\ r_3 & 0 & -r_1 \\ -r_2 & r_1 & 0 \end{bmatrix}$, we set up the following optimization problem

$$\arg\min_{\mathbf{r},\mathbf{v}} \sum_i ||((1 - \cos\theta_i)(\mathbf{r}\mathbf{r}^T - \mathbf{I}) + \sin\theta_i\mathbf{Q}_\mathbf{r})\mathbf{v} - \mathbf{t}_i|| \tag{D.3}$$

$$\text{s.t.} \quad ||\mathbf{r}|| = 1. \tag{D.4}$$

The above optimization problem is non-convex with respect to $\mathbf{r}, \mathbf{v}$. Since the rotation axis $\mathbf{r}$ has only 2 degrees of freedom we use spherical coordinates (ψ, ξ) to

Figure D.5: Match cost for pan (left) and tilt (right) rotation.

parameterize $\mathbf{r}$

$$
\begin{aligned}
r_1 &= \cos\psi \sin\xi \\
r_2 &= \sin\psi \sin\xi \\
r_3 &= \cos\xi
\end{aligned}
$$

and then solve the convex optimization problem with respect to $\mathbf{v}$. Fig. D.5 displays the minimum cost that we obtained for different angles ψ, ξ. We display the solution to the optimization for pan and tilt angles in Table D.2.

D.2.2 Estimating the rotation of the camera coordinate system (ω)

The rotation axis measurements are displayed on the second column of Table D.1. Notice that the first five rotation vectors are approximately the opposite of the last five. This is expected since the angle of the rotation is reversed. Since the vectors are consistent, instead of formulating and solving a complex non-convex optimization problem, we estimate the rotation axis with a simple average operation. More specifically, we compute the average value for the two variables and use the

Table D.2: Calibration Results

	PTU Pan			PTU Tilt		
rotation axis $\mathbf{r}$	-0.052	0.997	-0.052	-0.940	0.324	-0.105
vector $\mathbf{v}$	46.554	312.956	-3.629	-221.493	-86.002	-2.419
matching cost		9.12			4.10	
rotation vector ω	0.053	0.997	-0.053	-0.994	0.110	-0.016

contraint that the norm of the rotation vector is one to obtain the value for the third coordinate. The results for both pan and tilt are displayed in Table D.2.

D.3 Computing the external parameters for any PTU rotation

To synopsize the calibration process, here are the equations that provide the camera translation $\mathbf{T}$ and rotation $\mathbf{R}_\omega$ as a function of the pan and tilt of the PTU. We denote with ϕ, θ the pan and tilt angle of the PTU respectively.

$$\begin{aligned}\mathbf{T}(\phi) \quad = \quad & ((1-\cos\phi)\cdot\begin{bmatrix} -0.997 & -0.052 & 0.003 \\ -0.052 & -0.006 & -0.052 \\ 0.003 & -0.052 & -0.997 \end{bmatrix} + \\ & \sin\phi\cdot\begin{bmatrix} 0 & 0.052 & 0.997 \\ -0.052 & 0 & 0.052 \\ -0.997 & -0.052 & 0 \end{bmatrix})\cdot\begin{bmatrix} 46.554 \\ 312.956 \\ -3.629 \end{bmatrix}\end{aligned} \tag{D.5}$$

$$\mathbf{R}_{\omega_\phi}(\phi) = \begin{bmatrix} 0.003 & 0.053 & -0.003 \\ 0.053 & 0.994 & -0.053 \\ -0.003 & -0.053 & 0.003 \end{bmatrix} + \begin{bmatrix} 0.997 & -0.053 & 0.003 \\ -0.053 & 0.006 & 0.053 \\ 0.003 & 0.053 & 0.997 \end{bmatrix} \cdot \cos\phi + \begin{bmatrix} 0 & 0.053 & 0.997 \\ -0.053 & 0 & -0.053 \\ -0.997 & 0.053 & 0 \end{bmatrix} \cdot \sin\phi \tag{D.6}$$

$$\mathbf{T}(\theta) = ((1-\cos\theta) \cdot \begin{bmatrix} -0.116 & -0.304 & 0.099 \\ -0.304 & -0.896 & -0.034 \\ 0.099 & -0.033 & -0.989 \end{bmatrix} + \sin\theta \cdot \begin{bmatrix} 0 & 0.105 & 0.323 \\ -0.105 & 0 & 0.940 \\ -0.323 & -0.940 & 0 \end{bmatrix}) \cdot \begin{bmatrix} -221.493 \\ -86.002 \\ -2.419 \end{bmatrix} \tag{D.7}$$

$$\mathbf{R}_{\omega_\theta}(\theta) = \begin{bmatrix} 0.988 & -0.110 & 0.016 \\ -0.110 & 0.012 & -0.002 \\ 0.016 & -0.002 & 0.000 \end{bmatrix} + \begin{bmatrix} 0.012 & 0.110 & -0.016 \\ 0.110 & 0.988 & 0.002 \\ -0.016 & 0.002 & 1 \end{bmatrix} \cdot \cos\theta + \begin{bmatrix} 0 & 0.016 & 0.110 \\ -0.016 & 0 & 0.994 \\ -0.110 & -0.994 & 0 \end{bmatrix} \cdot \sin\theta \tag{D.8}$$

Bibliography

[1] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," pp. 15–18, 2006.

[2] Z.-F. Wang and Z.-G. Zheng, "A region based stereo matching algorithm using cooperative optimization," *CVPR*, pp. 1–8, 2008.

[3] S. Borra and S. Sarkar, "A framework for performance characterization of intermediate level grouping modules," *PAMI*, vol. 19, no. 11, pp. 1306–1312, 1997.

[4] P. Monasse and F. Guichard, "Fast computation of a contrast-invariant image representation," *IEEE Trans. on Image Processing*, vol. 9, no. 5, pp. 860–872, 2000.

[5] S. Paris and F. Durand, "A topological approach to hierarchical segmentation using mean shift.," *CVPR*, 2007.

[6] B. Hong, K. Ni, S. Soatto, and T. Chan, "Unsupervised multiphase segmentation: a recursive approach," *Computer Vision and Image Understanding*, April 2009.

[7] P. Felzenszwalb and D. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, pp. 167–181, 2004.

[8] S. Alpert, M. Galun, R. Basri, and A. Brandt, "Image segmentation by probabilistic bottom-up aggregation and cue integration," *CVPR*, pp. 1–8, 2007.

[9] K. Fukunaga and L. Hostetler, "The estimation of the gradient of a density function with applications in pattern recognition," *IEEE Trans. Information Theory*, vol. 21, pp. 32–40, 1975.

[10] Y. Cheng, "Mean shift, mode seeking, and clustering," *PAMI*, vol. 17, pp. 790–799, 1995.

[11] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. on PAMI*, pp. 603–619, 2002.

[12] S. Smith and J. Brady, "Susan a new approach to low level image processing," *IJCV*, vol. 23, pp. 45–78, 1997.

[13] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," *ICCV*, pp. 839–846, 1998.

[14] J. van de Weijer and R. van den Boomgaard, "Local mode filtering," *CVPR*, vol. 2, pp. 428–432, 2001.

[15] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *PAMI*, vol. 12, no. 7, pp. 629–639, 1990.

[16] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *PAMI*, vol. 25, pp. 564– 577, 2003.

[17] Y. Wei and L. Quan, "Region-based progressive stereo matching," *CVPR*, pp. 106–113, 2004.

[18] B. Georgescu, I. Shimshoni, and P. Meer, "Mean shift based clustering in high dimensions: A texture classification example," *ICCV*, pp. 456–463, 2003.

[19] D. DeMenthon and R. Megret, "Spatio-temporal segmentation of video by hierarchical mean shift analysis," tech. rep., 2002.

[20] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," *ICCV*, pp. 464–471, 2003.

[21] M. Fashing and C. Tomasi, "Mean shift is a bound optimization," *PAMI*, vol. 27, pp. 471–474, 2005.

[22] X. Yuan and S. Z. Li, "Half quadratic analysis for mean shift: with extension to a sequential data mode-seeking method," *Computer Vision, IEEE International Conference on*, vol. 0, pp. 1–8, 2007.

[23] M. Carreira-Perpinan, "Gaussian mean-shift is an em algorithm," *IEEE Trans. PAMI*, vol. 29, no. 5, pp. 767–776, 2007.

[24] C. Shen, M. Brooks, and A. Hengel, "Fast global kernel density mode seeking: Applications to localization and tracking.," *IEEE Transactions on Image Processing*, vol. 16, no. 5, pp. 1457–1469, 2007.

[25] R. Subbarao and P. Meer, "Nonlinear mean shift for clustering over analytic manifolds," *CVPR*, pp. 1168 – 1175, 2006.

[26] Y. Sheikh, E.Khan, and T. Kanade, "Mode-seeking by medoidshifts," *ICCV*, October 2007.

[27] A. Vedaldi and S. Soatto, "Quick shift and kernel methods for mode seeking," *Proceedings of the European Conference on Computer Vision*, 2008.

[28] R. Subbarao and P. Meer, "Nonlinear mean shift over riemannian manifolds," *IJCV*, vol. 84, no. 1, pp. 1 – 20, 2009.

[29] V. Epanechnikov, "Nonparametric estimation of a multivariate probability density," *Theory Prob. Appl. (USSR)*, vol. 14, pp. 153–158, 1969.

[30] S. Rao, A. Martins, and J. Principe, "Mean shift: An information theoretic perspective," *Pattern Recognition Letters*, vol. 30, no. 3, pp. 222 – 230, 2009.

[31] M. Black, G. Sapiro, D. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 7, no. 3, pp. 421–432, 1998.

[32] P. Raghu, R. Poongodi, and B. Yegnanarayana, "Unsupervised texture classification using vector quantization and deterministic relaxation neural network," *IEEE Transactions on Image Processing*, vol. 6, no. 10, pp. 1376–1387, 1997.

[33] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *IJCV*, vol. 43, no. 1, pp. 7–27, 2001.

[34] D. Defour, F. D. Dinechin, and J. Muller, "A new scheme for table-based evaluation of functions," Tech. Rep. ISSN 0249-6399, Institut National de Recherche en Informatique et en Automatique, 2002.

[35] C. Christoudias, B. Georgescu, and P. Meer, "Synergism in low-level vision," *ICPR*, vol. 4, pp. 150–155, August 2002.

[36] 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," *ICCV*, vol. 2, pp. 416–423, 2001.

[37] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "A measure for objective evaluation of image segmentation algorithms," *Workshop on Empirical Evaluation Methods in Computer Vision, CVPR*, 2005.

[38] M. Meila, "Comparing clusterings by the variation of information," *Conference Learning Theory*, 2003.

[39] M. Meila, "Comparing clusterings: an axiomatic view," *ICML*, pp. 577 – 584, 2005.

[40] R. Unnikrishnan, C. Pantofaru, and M. Hebert, "Toward objective evaluation of image segmentation algorithms," *PAMI*, vol. 29, no. 6, pp. 929–944, 2007.

[41] J. Freixenet, X. Munoz, D. Raba, J. Marti, and X. Cuff, "Yet another survey on image segmentation: Region and boundary information integration," *ECCV*, pp. 408–422, 2002.

[42] A. Y. Yang, J. Wright, Y. Ma, and S. Sastry, "Unsupervised segmentation of natural images via lossy data compression," *Comput. Vis. Image Underst.*, vol. 110, no. 2, pp. 212–225, 2008.

[43] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[44] O. D. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[45] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys, "Towards urban 3d reconstruction from video," *3DPVT*, 2006.

[46] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium* (G. Lakemeyer and B. Nebel, eds.), Morgan Kaufmann, 2002.

[47] A. Davison, "Real-time simultaneous localization and mapping with a single camera," *ICCV*, pp. 1403–1410, 2003.

[48] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation and bundle adjustment," *Computer Vision and Image Understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[49] J. Weng, T. Huang, and N. Ahuja, "Motion and structure from line correspondences," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 3, pp. 318–336, 1992.

[50] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *Proc. British Machine Vision Conference*, (Edinburgh), pp. 17–26, 2006.

[51] G. Desouza and A. Kak, "Vision for mobile robot navigation: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 237–267, February 2002.

[52] M. Srinivasan, J. Chahl, K. Weber, S. Venkatesh, M. Nagle, and S. Zhang, "Robot navigation inspired by principles of insect vision," *Robotics and Autonomous Systems*, vol. 26, no. 2, pp. 203–216, 1999.

[53] D. Coombs and K. Roberts, "Centering behavior using peripheral vision," *CVPR*, pp. 440–445, 1993.

[54] D. Coombs, M. Herman, T. Hong, and M. Nashman, "Real-time obstacle avoidance using central flow divergence and peripheral flow," *ICCV*, pp. 276–283, 1995.

[55] A. Duchon and W. Warren, "Robot navigation from a gibsonian viewpoint," *IEEE Conference on Systems, Man and Cybernetics*, pp. 2272–2277, 1994.

[56] J. Santos-Victor, G. Sandini, F. Curotto, and S. Garibaldi, "Divergent stereo for robot navigation : Learning from bees," *CVPR*, pp. 434–439, 1993.

[57] A. Argyros and F. Bergholm, "Combining central and peripheral vision for reactive robot navigation," *CVPR*, vol. 2, pp. 646–651, 1999.

[58] A. Argyros, D. Tsakiris, and C. Groyer, "Biomimetic centering behavior: Mobile robots with panoramic sensors," in *IEEE Robotics and Automation Magazine , special issue on Panoramic Robotics* (K. Daniilides and N. Papanikolopoulos, eds.), vol. 11, Dec 2004.

[59] B. P. Bogert, M. J. R. Healy, and J. W. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum, and saphe cracking," in *Time Series Analysis* (M. Rosenblatt, ed.), pp. 209–243, 1963.

[60] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[61] R. W. S. L. R. Rabiner, *Digital Processing of Speech Signals.* Prentice Hall, 1978.

[62] B. Srinivasa and B. N. Chatterji, "An fft-based technique for translation, rotation and scale-invariant image registration," *IEEE Transactions on Image Processing*, vol. 8, no. 8, pp. 1266–1271, 1996.

[63] C. Kuglin and D. Hines, "The phase correlation image alignment method," *IEEE Conference on Cybernetics and Society*, pp. 163–165, 1975.

[64] H. Foroosh, J. B. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," *IEEE Transactions on Image Processing*, vol. 11, pp. 188–200, 2002.

[65] D. Fleet and A. Jepson, "Computation of component image velocity from local phase information," *IJCV*, vol. 5, pp. 77–104, 1990.

[66] H. Ji and C. Fermüller, "Noise causes slant underestimation in stereo and motion," *Vision Research*, vol. 46, no. 19, pp. 3105–3120, 2006.

[67] K. Daniilidis, *On the Error Sensitivity in the Recovery of Object Descriptions.* PhD thesis, Department of Informatics, University of Karlsruhe, Germany, 1992. In German.

[68] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

[69] E. Simoncelli, "Design of multi-dimensional derivative filters," *ICIP*, vol. 1, pp. 790–793, 1994.

[70] R. Y. Tsai and T. S. Huang, "Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces," *Trans. on PAMI*, vol. 6, pp. 13–27, 1984.

[71] T. Brodsky, C. Fermüller, and Y. Aloimonos, "Structure from motion: Beyond the epipolar constraint," *Int. J. Computer Vision*, vol. 37, pp. 231–258, 2000.

[72] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 652–659, 2004.

[73] H. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proc. Royal Society of London*, vol. 208, no. B, pp. 385–397, 1980.

[74] H. Longuet-Higgins, "A computer program for reconstructing a scene from two projections," *Nature*, vol. 293, no. 11, pp. 133–135, 1981.

[75] A. Waxman and K. Wohn, "Contour evolution, neighborhood deformation, and global image flow: planar surfaces in motion.," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 95–108, 1985.

[76] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. New Jersey: Prentice Hall, 1998.

[77] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, *Bundle Adjustment – A Modern Synthesis*, vol. 1883. Springer Verlag, January 2000.

[78] C. Engels, H. Stewenius, and D. Nister, "Bundle adjustment rules," *Photogrammetric Computer Vision*, 2006.

[79] G. Adiv, "Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field," *PAMI*, vol. 11, pp. 477–489, 1989.

[80] K. Daniilidis and H. Nagel, "The coupling of rotation and translation in motion estimation of planar surfaces," *CVPR*, pp. 188–193, 1993.

[81] K. Daniilidis and M. Spetsakis, *Understanding noise sensitivity in structure from motion*. Lawrence Elrlbaum Associates, 1997.

[82] R. Dutta and M. Snyder, "Robustness of correspondence-based structure from motion," *ICCV*, pp. 106–110, 1990.

[83] J. Weng, T. Huang, and N. Ahuja, *Motion and Structure from Image Sequences*. Springer-Verlag, 1991.

[84] G. Young and R. Chellappa, "Statistical analysis of inherent ambiguities in recovering 3-d motion from a noisy flow field," *PAMI*, vol. 14, pp. 995–1013, 1992.

[85] L. Cheong, C. Fermüller, and Y. Aloimonos, "Interaction between 3d shape and motion: Theory and applications," tech. rep., CVL, Univ. of Maryland, 1996.

[86] C. Baillard and A. Zisserman, "Automatic reconstruction of piecewise planar models from multiple views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 559–565, June 1999.

[87] A. Dick, P. Torr, and R. Cipolla, "Automatic 3d modelling of architecture," *British Machine Vision Conference*, pp. 372–381, 2000.

[88] E. Vincent and R. Laganiere, "Detecting planar homographies in an image pair," in *Proc. 2nd International Symposium on Image and Signal Processing and Analysis ISPA 2001*, pp. 182–187, 2001.

[89] T. Werner and A. Zisserman, "New techniques for automated architectural reconstruction from photographs," pp. 541–555, 2002.

[90] T. Werner and A. Zisserman, "Model selection for automated architectural reconstruction from multiple views," *Proceedings of the British Machine Vision Conference*, pp. 53–62, 2002.

[91] K. Schindler, *Generalized Use of Homographies for Piecewise Planar Reconstruction*, vol. 2749 of *Lecture Notes in Computer Science*, pp. 276–284. Berlin: Springer, 2003.

[92] G. Wang, H. Tsui, and Z. Hu, "Reconstruction of structured scenes from two uncalibrated images," *Pattern Recogn. Lett.*, vol. 26, no. 2, pp. 207–220, 2005.

[93] A. Amintabar and B. Boufama, "Homography-based plane identification and matching," in *Proc. 15th IEEE International Conference on Image Processing ICIP 2008*, pp. 297–300, 2008.

[94] H. Tao, H. Sawhney, and R. Kumar, "A global matching framework for stereo computation," in *Proc. Eighth IEEE International Conference on Computer Vision ICCV 2001*, vol. 1, pp. 532–539 vol.1, 2001.

[95] A. Criminisi, I. Reid, and A. Zisserman, "A plane measuring device," *Image and Vision Computing*, vol. 17, no. 8, pp. 625 – 634, 1999.

[96] D. Marr, *Vision*. W H Freeman, 1982.

[97] R. Bajcsy, "Active perception," *Proc. of the IEEE special issue on Computer Vision*, vol. 76, pp. 966–1005, August 1988.

[98] J. Y. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," *International Journal of Computer Vision*, vol. 1, pp. 333–356, 1988.

[99] D. H. Ballard, "Animate vision," *Artificial Intelligence*, vol. 48, pp. 57–86, 1991.

[100] J. Aloimonos, "Purposive and qualitative active vision," *ICPR*, pp. 346–360, 1990.

[101] Y. Aloimonos, ed., *Active Perception*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1993.

[102] C. Fermüller and Y. Aloimonos, "Vision and action," *Image and Vision Computing*, vol. 13, no. 10, pp. 725–744, 1995.

[103] H. Christensen and J. Eklundh, "Active vision from multiple cues," *IEEE International Workshop on Biologically Motivated Computer Vision*, pp. 209–216, 2000.

[104] S. Soatto, "Actionable information in vision," *ICCV*, October 2009.

[105] K. Kutulakos and C. Dyer, "Recovering shape by purposive viewpoint adjustment," *CVPR*, pp. 16–22, 92.

[106] E. Dickmanns, *Dynamic Vision for Perception and Control of Motion*. Springer, 2007.

[107] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie Mellon University, April 1991.

[108] J. Shi and C. Tomasi, "Good features to track," *CVPR*, pp. 593–600, 1994.

[109] M. Fischler and R. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[110] P. Besl and N. McKay, "A method for registration of 3-d shapes," *PAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[111] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image Vision Comput.*, vol. 10, no. 3, pp. 145–155, 1992.

[112] T. D. Sanger, "Stereo disparity computation using gabor filters," *Biological Cybernetics*, vol. 59, pp. 405–418, 1988.

[113] M. Jenkin and A. D. Jepson, "The measurement of binocular disparity," in *Computational Processes in Human Vision* (Z. Pylyshyn, ed.), (New Jersey), pp. 69–98, Ablex Press, 1988.

[114] A. Jepson and D. Fleet, "Phase singularities in scale-space," *Image and Vision Computing Journal*, vol. 9, no. 5, pp. 338–343, 1991.

[115] D. Fleet and A. Jepson, "Stability of phase information," *IEEE Trans on PAMI*, vol. 15, no. 12, pp. 1253–1268, 1993.

[116] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt, "Performance of optical flow techniques," *Int. J. Comput. Vision*, vol. 12, no. 1, pp. 43–77, 1994.

[117] A. S. Ogale and Y. Aloimonos, "Stereo correspondence with slanted surfaces: critical implications of horizontal slant," *CVPR*, vol. 1, pp. 568–573, 2004.

[118] L. Cohen, "The scale representation," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3275–3292, 1993.

[119] "Mayang's free textures." http://www.mayang.com/textures/.

[120] J. Weng, T. Huang, and N. Ahuja, "Motion and structure from two perspective views: Algorithms, error analysis, and error estimation," *PAMI*, vol. 11, no. 5, pp. 451–476, 1989.

[121] J. Y. Bouguet, "Camera calibration toolbox for matlab," 2008.