Segmentation of Image Sequence into Scene-Coherent Parts

Johan Sjöberg

January 7, 2015

Abstract

The Narrative Clip is a small wearable camera that takes an image every 30 seconds. By wearing the clip a whole day a user captures a long image sequence of the day's events. In this thesis we will segment such a sequence into the individual events automatically.

Multiple sequences are segmented by humans in order to find a groundtruth for each sequence. The groundtruth will be used to determine the performance of the algorithm and also how well individual humans are able to segment a sequence.

The method presented here takes a sequence of images and tries to find the location where the mean of the descriptors changes. The images are described using various image descriptors that capture colors, lines, textures and similar low level features. We also introduce an indoor/outdoor classification method that combines a SVM and a HMM. The classification method is combined with the segmentation for each descriptor in order to create a combined segmentation.

The indoor classification method achieves an accuracy of 97% which is to be considered very good results. The best human segmentation has an F1-score of 0.82 while the best automatic segmentation method's F1-score is 0.43. The conclusion is that the current system is not suitable for any practical usage.

Preface

This report is my master thesis for the degree in Master of Science in Engineering Physics at Lund Institute of Technology (LTH). It was conducted at Narrative AB in Lund during 2014 with the help of the Mathematical Image Group at the Centre for Mathematical Science, LTH. It was performed by me alone with help from my supervisor at Narrative, Daniel Hamngren and my two supervisors at LTH, Kalle Åström and Cristian Sminchisescu.

I would like to take this opportunity to thank all of my supervisors for their support and guidance.

Contents

1 Introduction

Recent developments in consumer electronics have sparked a huge interest in wearable technology. As the name suggests this branch consists of products that in one way or another are attached to someones clothes or body and carried through out the day. Multiple products have already reached the market and the epithet is now describing a wide range of products, for example Google Glass, Motorola Moto 360, Fitbit Flex and Narrative Clip.

This category can be further divided into a "lifelogging" segment for products with the purpose to collect information during a user's day. One such product is the Narrative Clip, a small camera that takes one image every 30 seconds with the purpose of creating a photographic memory. The camera is attached to a shirt and will automatically takes pictures during the user's day. The images are later uploaded to the Narrative Cloud Service and divided into moments which lets the user review segments of their day.

1.1 Aim of the thesis

Using the Narrative Clip generates a huge amount of images, a full day can result in over 1500 images, few users have the time to even scroll through that amount of images each day. In order to make the product useful the data has to be automatically both limited and structured to only show the essential parts of a day. A natural approach, for an algorithm like that, would be to automatically divides each day into the separate moments or events that characterize the day. This is actually already being done today but unfortunately with various amount of success. A reliable segmentation would be very beneficial and make it possible to calculate a moment's class or relevance. Information crucial when searching for a moment or deciding which moments that should be displayed. The goal of this thesis is to further develop this segmentation that automatically divide a day into moments.

This segmentation will be done by reviewing multiple approaches for scene recognition and incorporating these features into a time-series segmentation algorithm. Training data have been collected throughout the thesis work providing means to both train and evaluate the result of the algorithm. The goal is to create a system that outperforms the current one and provides a start for a system to classify moments.

1.2 Moment definition

A moment is a rather loose concept, it perfectly defines both playing soccer as well as kicking the ball. In this context the correct definition is playing football due to our interest in collecting numerous images that are 30 seconds apart. This limit our definition of a moment to a certain time resolution, a moment is neither too short nor is it too long. It is troublesome to define an exact range since the limit is both subjective as well as dependent on the variance of a day. This problem combined with that the number of different moments are virtually infinite prevents us from explicitly defining a moment. Instead we embrace a simplified but still somewhat vague definition of a moment:

Definition. A moment is a sequence of time-consecutive images that are visually coherent in the sense that they have similar color, texture or other characteristics. The length of the sequence is in proportion to the sampling rate and thus allows for a certain amount of outliers in the sequence.

This definition is vague in the sense of coherence and length but it is implementable if we allow for some leeway in these parameters. Not only is the definition vague but it also comes with limitations, for example it could be difficult to differentiate between playing soccer and American football since both are played outside in the grass. Luckily most of these limitations are avoided due to events like these being separated in time, for example most of the times you don't start playing American football right after a football practice.

If two images have the same color mean that would indicate that the two images are similar or coherent. Extending this reasoning to moments would imply that the images in the moment must have a somewhat constant mean in order to be considered coherent. Humans use the same approach but at a much higher abstraction level than simple colors. A constant mean for a human might be interpreted as doing the same thing even though the colors changes.

1.3 Constraints

The whole thesis should be viewed as a proof of concept rather than an optimization problem. This results in a thesis that will be restricted in multiple ways.

- Limited dataset, the datasets will not be representative for the average user. Although the total number of images will be rather large, the number of days will be limited. This is a result of computational constraints as well as the time-consuming work of collecting multiple days with various settings.
- Limited number of people have segmented the data, a real project would require a lot more people to get a good statistical foundation.
- Various computational limitations, such memory and processing power. The work has been done on a 16-core machine with plenty of memory. Yet there exists situations when this is not enough, in particular when using the k-means clustering algorithm or training the SVM classification algorithm.

- Time limitations, the thesis is a time-limited project and multiple methods and algorithms have been ignored due to time-constraints.
- Any extra image data, for example the GPS-data, have been ignored due to various circumstances.

1.4 Thesis structure

This thesis will outline an algorithm to segment a sequence of images into moments. We start by introducing the Narrative Clip and the images that the algorithm was tested on. This is followed up with the theory section 3 where we introduce all the methods used in our algorithms. The methods are further divided into different subsections, the most important being the feature subsection 3.5 where we introduce the features used in this thesis. Other important subsections here are the classification subsection 3.6, the changepoint subsection 3.7 and the evaluation methods subsection 3.8.

The algorithms presented in the method section are used in the following three sections. In the indoor/outdoor classification section 4 we explain how to classify our sequences as either indoors or outdoors. In the segmentation strategies section 5 we introduce our way to segment different features separately as well as combined. In the evaluation section 6 we show the results of our classification and segmentation algorithms.

Finally we discuss all the different results in the discussion section 7 and conclude how the results could be improved in the conclusion section 8.

2 Setup

2.1 Narrative Clip

The Narrative clip is a small camera, it weights about 20 gram and is only 36x36x9 mm large. In addition to the five megapixel camera the clip also has a GPS, magnetometer and an accelerometer. In this thesis we will only use the images even though the extra information in the GPS could be useful. The size of the camera allow a user to clip it onto a shirt and wear it a whole day without any obstruction or thought.

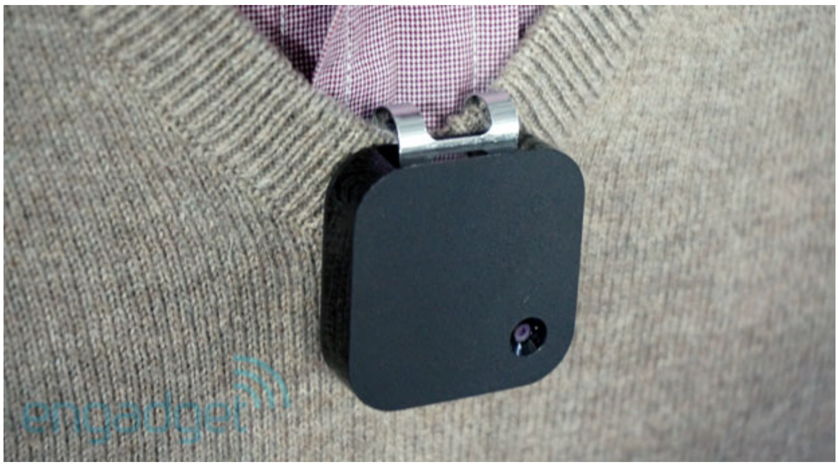

Figure 1: A narrative clip attached to a shirt.

The camera takes one image every 30 seconds resulting in well over 1500 images if worn a full day. The images are stored on the camera and later uploaded via a computer to the cloud. No image processing is done in the camera or on the users computer other than what is needed when taking and storing an image. The images are stored using the Narrative Cloud Service (NCS) which allows mobile access using the Narrative app. All the image processing is performed using the NCS, this includes three processes that are particularly relevant for this thesis.

- The images are rotated using the accelerometer data, a process that minimizes problems caused by rotations.
- All images are resized which is a crucial task in order to run our algorithms quickly.
- All bad images are removed, which includes images with that are too dark or has too much motion blur. This process removes extreme outliers that might cause problems with our algorithms.

The NCS is also used for other major image processing tasks that is not relevant for this thesis, like the current method segment a sequence into moments. For a visual representation of the flow, see figure 2.

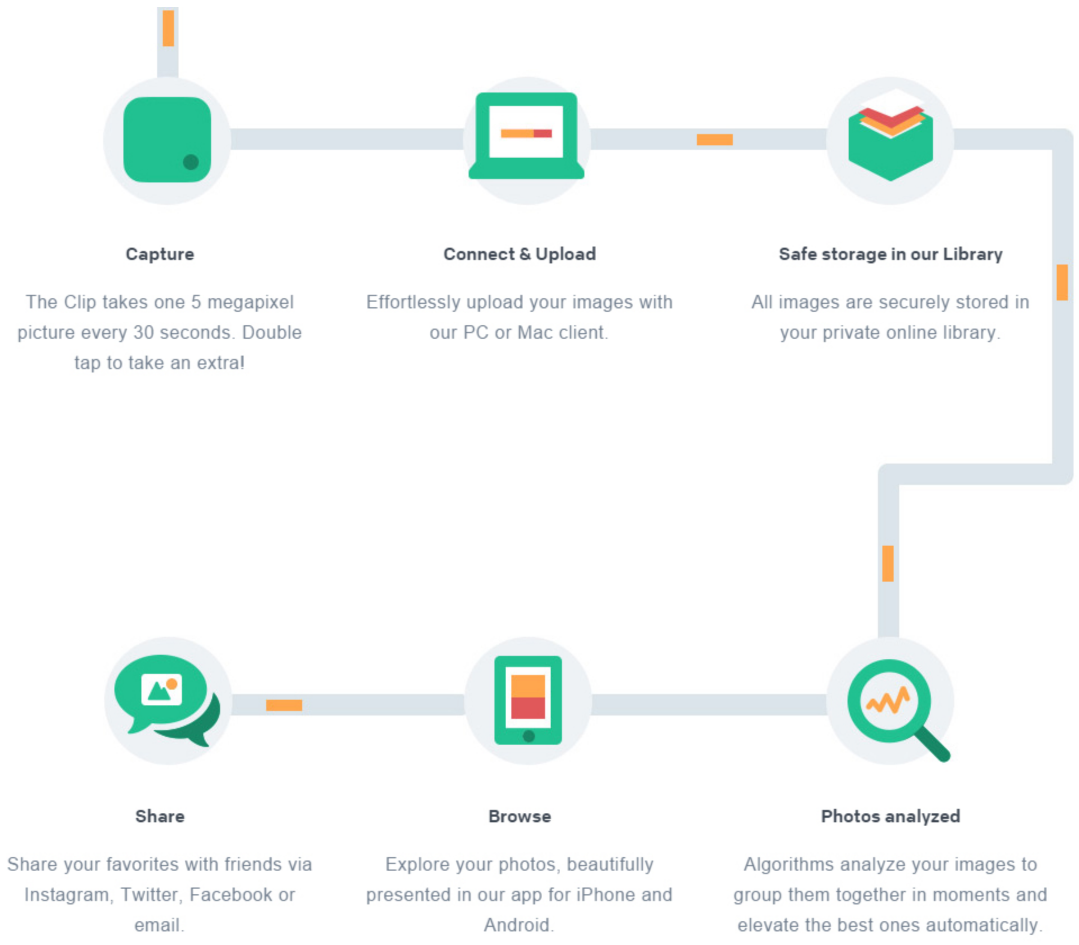
Capture
The Clip takes one 5 megapixel picture every 30 seconds. Double tap to take an extra!
Connect & Upload
Effortlessly upload your images with our PC or Mac client.
Safe storage in our Library
All images are securely stored in your private online library.
Photos analyzed
Algorithms analyze your images to group them together in moments and elevate the best ones automatically.
Browse
Explore your photos, beautifully presented in our app for iPhone and Android.
Share
Share your favorites with friends via Instagram, Twitter, Facebook or email.

Figure 2: A flow chart of how the Narrative system works.

Since no image processing is done in real-time the algorithms are fairly unconstrained. This means that they can use all the images in a sequence and they don't need to conform to a particular performance requirement, although faster is better. This is why no particular interest have been given to the speed of any algorithms in this thesis. This also why the algorithm presented here should be viewed as a proof of concept rather than a working algorithm.

2.2 Images

All images used in this thesis have been acquired using the Narrative clip. At the start of the thesis no suitable dataset existed that captured multiple days with a time-resolution and perspective similar to that of the Narrative clip. Instead we created a smaller set consisting of about 34 longer photo-sets which sums to about 24 000 images in total. The photo-sets aren't necessarily complete days but they are guaranteed to be more than two hours apart. The images were captured by two persons during the spring and summer at different locations in Sweden and Europe. The images were mostly captured in situations where most people would use the Narrative clip. For example, it is more likely to use the clip when meeting your friends rather than sitting in front of a computer. We tried to capture this behavior in our dataset but it is fairly safe to say that the clip will be used in a lot more varying situations than our test data. Although the variety in the dataset is limited it should be suitable for a proof of concept.

Dividing a sequence into smaller pieces is a highly subjective procedure and even for a human a rather hard problem. It is in particular hard to determine the resolution of a moment, for example consider the following actions: "Walking to the car", "Driving the car" and "Walking to the apartment". Should a segmentation be split into the three different actions or should they be merged into one moment describing the complete transportation? In order to find the "optimal" answer and limit the subjective reasoning we had more than one person segment the data. This is however a time consuming task and would require a lot of people to be done properly. The photo-sets was therefor, for the most parts, segmented by two people with a good understanding of the Narrative Clip. However three photo-sets were chosen to be segmented more extensively. These photo-sets were segmented by four people with a good understanding of the Narrative clip and 8 people with minor or no understanding of the Narrative Clip. The three photo-sets will be referred to as "Summerday", "Night festival" and "Golf". We will in particular pay attention to the "Summerday" photoset.

The different segmentations where combined for each sequence using the *sequence averaging method* described in section 3.1. This results in one single segmentation per photoset and will be referred to as the groundtruth segmentation. This sequence is considered as the correct one and is the primary sequence that the segmentation algorithm will be compared against.

In this thesis an indoor/outdoor classification algorithm will be presented. This algorithm has also been trained using the same images above. For this application we annotated all the images as either indoor or outdoors. It turned out that approximately 52% of the images were taken indoors thus confirming that the clip is used everywhere. This annotated set was divided into two random subsets. A training set with 40% of the images and the test set with the remaining 60% of the images.

3 Theory

3.1 Sequence averaging

Consider a sequence $X = X_1, X_2, ..., X_N$ that has been segmented into U number of segments. Each location where the sequence has been segmented is called a segmentation point, we will have U-1 segmentation points in a segmentation. Segmenting a sequence can be done in multiple ways with a different number of segmentation points and with different segmentation point locations. Consider M such different ways. We are interested in finding a complete segmentation of X that combine the M different ways into one optimal segmentation. We do not know the number of segments in the complete segmentation, everything has to be inferred using M segmentations. In this approach we are interested in finding the locations where the majority of the segmentations have segmentation points that agree with each other. Different segmentations might not agree with each other perfectly which is why we allow for some offset.

Consider that each X is replaced with a binary vector of the same length with ones before the location of a segmentation point and the rest zeros. Since we have M different segmentations we have M different binary vectors. As an example we use the following vectors:

$$M_1 : [01000010000000000100100000000]$$
$$M_2 : [01000100000001000000000010000]$$
$$M_3 : [01000000100000000100000000001]$$

By adding these vectors together we get a new vector with a maximum value of M in those locations where all the binary vectors agree with each other.

$$Summed : [03000111000010002001000100001]$$

By sliding a window and summing the content inside the window we create a vector that allows for some offset in the segmentation points. In this example we use a window with a length of three.

$$Windowed : [33301232100111022211101111011]$$

We find the local maximums or plateaus of this vector but add the requirement that these have to be above a certain threshold and further apart than the width of the sliding window. In our example we require the threshold to be above one to ensure we select only the segmentation points where the majority agree with each other.

$$Thresholded : [03000030000000000200000000000]$$

All the points that are not zero are the most relevant points for these particular M segmentations and will be considered segmentation points.

3.2 Color space models

An image is a mathematical representation of the real world seen through a camera. For this representation to be useful we need a transformation from the physical world to the mathematical. One such transformation could be to represent an image using the different wavelengths. Multiple color spaces have been created since different applications have different needs. Some are focusing on the human perception while other take a more physical approach. Here we present the color spaces that have been used in this thesis:

3.2.1 RGB

One of the most commonly used color spaces is the RGB space. The idea is to use different amount of red, green and blue light in order to create various colors. This concept is based on the human visual system, a human eye has three types of cones, each sensitive to different wavelength corresponding to blue, green and red. By shining at the cones with three different colors we stimulate them differently and a certain color can be experienced. The RGB model can be represented with a Cartesian coordinate system, similar to figure 3. Each axis represent the intensity of red, green and blue thus creating a color cube.

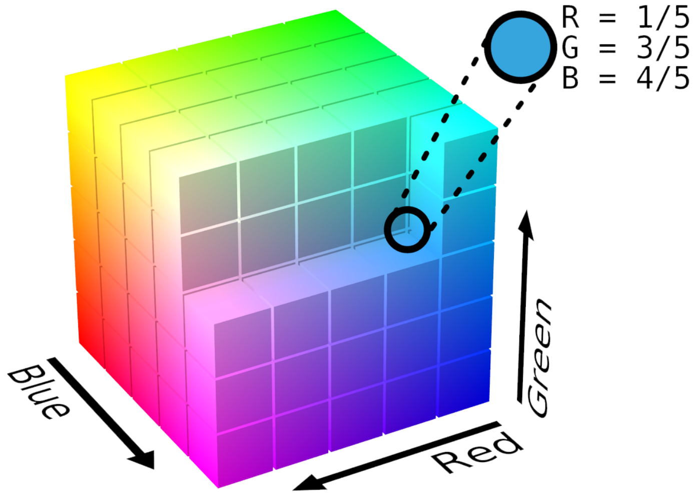

Figure 3: An illustration of the RGB colorspace.

The RGB is straight forward with clear connection to the human visual system but it is not without problems. The largest one being that it is not perceptually linear. For example, humans are more sensitive to green colors [3]. The RGB space doesn't reflect this and thus using the euclidean distance between two colors might not be the same as the perceived distance. This can cause problems in scene recognition due to the algorithm might prioritize, for human, nonessential details.

3.2.2 HSV

Few humans describe a color by adding different amounts of red, green and blue. Instead we tend to use descriptions like hue, saturation and brightness. With this in mind another very common color space was created, the Hue-Saturation-Value (HSV) color space. The HSV is a simple transformation of the RGB into a cylindrical representation. In this cylinder the angle describes a particular hue, the distance to the z-axis is the saturation and the height is the value or intensity. The strength of this representation is that the color is decoupled from the intensity. This makes it possible to select a color independently of the brightness, for example selecting a red apple is possible no matter what time of the day it is.

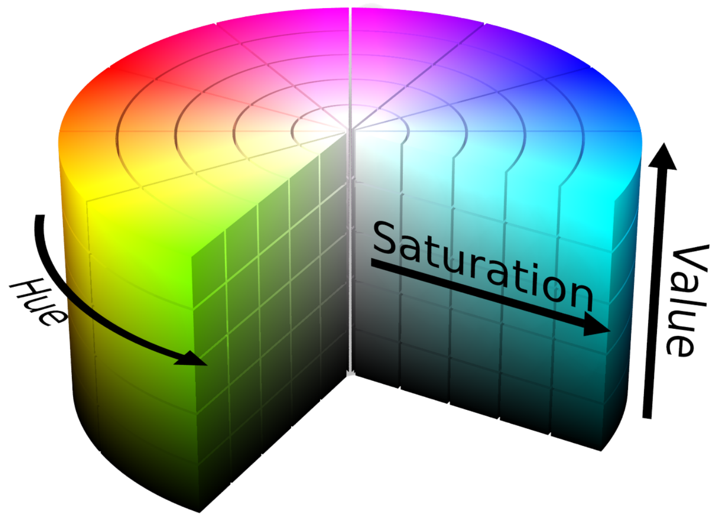

Figure 4: An illustration of the HSV colorspace.

The transformation from RGB to HSV preserves multiple properties of the RGB space, among those the perceived linearity problem. The HSV seem to solve the intensity problem of the RGB model but unfortunately it doesn't. The euclidean distance is still not equal to the perceived distance. For examples in figure 4, the yellow part seems brighter than the blue even though they have the same value. Gonzalez [4] describes it as ”perceived brightness is not a simple function of intensity”.

3.2.3 LAB

The CIELAB color space was created as a perceptually uniform color space. It is a perceptually linear color space in the sense that a change in one color doesn't necessarily alter the intensity in the same way as a change in another color. Instead it is the perceived change that stay the same. These properties make the LAB suitable when measuring the euclidean distance between images. The measured distance between two images should be equal to the distance a human perceive.

3.2.4 Grayscale

Grayscale transformation strips the image of its colors and is a simplification of the real world. Since humans only experience colors there are no correct way to convert an image into grayscale. We could use either the red channel in RGB or lightness channel in CIEL*a*b. The generally accepted, and possible also the most natural, definition of grayscale is the human perception of intensity. In this thesis we'll be using the transformation

$$Y' = 0.299R' + 0.587G' + 0.114B'. \tag{1}$$

This is the gamma corrected luma channel and is the standard grayscale component of the PAL television encoding. It is a weighted sum of the RGB channel which takes into account that humans are more sensitive to green colors.

3.3 Principal Component Analysis

Principal component analysis (PCA) [19] is a statistical method that can be used to reduce the number of dimensions of a feature vector. It is a lossy method in the sense that data is removed in order to limit the dimensions and reduce the size of the data. The goal is to only save the "good" data and remove the useless and redundant parts. These parts are of course unwanted if we intend to decrease the data size. Most signals contain some level of noise and it is important that a dimension reduction saves the data and not the noise. By applying a transformation to the data using a set of linear uncorrelated variables that maximizes the variance for each dimension we get a maximum signal-to-noise ratio. A large signal-to-noise ratio is key to limit the effects of noise. Summarized we are looking for a set of linear uncorrelated variables that also maximizes the variance for each dimension. The first dimension, or principal component as it is called in PCA, has the most variance and the last the least. This transformation diagonalizes the covariance matrix since in a set of linear uncorrelated variables the covariance matrix is diagonal.

Our data is represented by X which is a $m \times n$ matrix. m is the number of features that we want to reduce and n is the number of observations used to find the principal components. The mean of the observations have been removed for each feature. We get the following problem, find a orthogonal matrix $\boldsymbol{P}$ that diagonalizes

$$\boldsymbol{S_Y} \equiv \frac{1}{n-1}\boldsymbol{YY}^T \tag{2}$$

where

$$\boldsymbol{Y} = \boldsymbol{PX} \tag{3}$$

thus resulting in

$$\boldsymbol{S_Y} = \frac{1}{n-1}\boldsymbol{PXX}^T\boldsymbol{P}^T. \tag{4}$$

$\boldsymbol{XX}^T$ can be proven to be a symmetric matrix. All symmetric matrices can be diagonalized as

$$\boldsymbol{XX}^T = \boldsymbol{EDE}^T. \tag{5}$$

Here $\boldsymbol{E}$ is a matrix where the columns are the eigenvectors of $\boldsymbol{XX}^T$ and $\boldsymbol{D}$ is the diagonalized matrix.

Substituting equation 5 into equation 4 we get

$$\boldsymbol{S_Y} = \frac{1}{n-1}\boldsymbol{PEDE}^T\boldsymbol{P}^T. \tag{6}$$

$\boldsymbol{E}^{-1} = \boldsymbol{E}^T$ since $\boldsymbol{E}$ is a orthogonal matrix. Thus by choosing $\boldsymbol{P} = \boldsymbol{E}^T$ in equation 6 we get

$$\boldsymbol{S_Y} = \frac{1}{n-1}\boldsymbol{E}^{-1}\boldsymbol{EDE}^{-1}\boldsymbol{E} \tag{7}$$

shortened to

$$\boldsymbol{S_Y} = \frac{1}{n-1}\boldsymbol{D}. \tag{8}$$

By making a smart choice of the transformation $\boldsymbol{P}$ we can prove that the covariance matrix will be diagonal. Thus we simply have to calculate the eigenvectors of $\boldsymbol{XX}^T$ to get the required transformation.

Figure 5 illustrates a transformation from a two-dimensional space into the new principal component space. As expected the first principal component has the most variance.

In practice the last dimensions usually have a very small amount of variance and are not contributing much when describing the image. Removing these dimensions would therefor not affect the overall representation very much and has the benefit that it reduces the size of the data. It can be viewed as a lossy compression method, the least relevant data is discarded

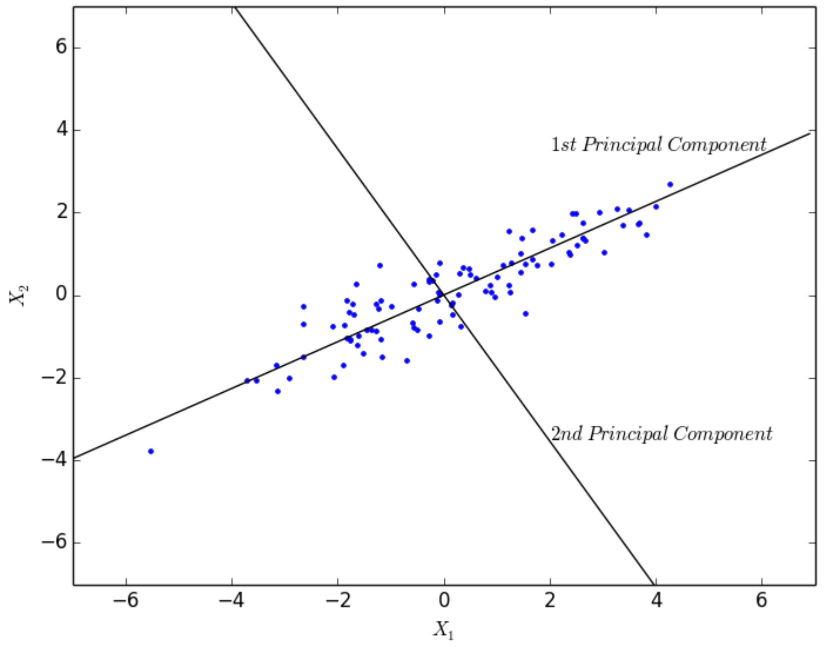

Figure 5: The two principal components of a two-dimensional space

in order to obtain a lower dimensional space. In figure 5 this would mean dropping the second principal component.

PCA requires a large training set in order to properly calculate the correct transformation. If the training data isn't complete risks are that the principal components are not the maximized variance at different levels. This might cause noise to be more pronounced and that we throw away critical data during the dimension reduction.

3.4 K-means clustering

Clustering is an essential non-supervised procedure that is used to find unknown structures in large amount of data. By looking at the similarity between multiple observations we can constructs groups, called clusters, that are all similar in a mathematical sense. For each cluster we can calculate a cluster center point that captures all the essential properties of that group. These cluster centers are usually a fictive point that is statistical relevant in some way. The points are fictive in the sense that they are not the same as any of the observations but rather created using the observations. Although the point usually is fictive it still understandable, for example clustering on multiple open landscapes images will generate a cluster center that, when reconstructed into an image, resemble an image of an open landscape.

K-means clustering [10] is a common clustering algorithm that tries to minimize the euclidean distance between the observations and the k cluster centers. Given the following $N \times d$ observations, $(\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N)$ and k sets, $(S_1, S_2, ..., S_k)$. The k-means objective is to find

$$\underset{\boldsymbol{S}}{argmin} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in S_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2. \tag{9}$$

To fulfill the objective the algorithm is comprised of two steps that are iterated until convergence, the assignment step and the update step. From a given starting set of cluster centers, $(\boldsymbol{m}_1^1, \boldsymbol{m}_2^1, ..., \boldsymbol{m}_k^1)$, the assignment step assigns the observations to the cluster center to which the euclidean distance is the smallest,

$$S_i^{(t)} = \{\boldsymbol{x_p} : \|\boldsymbol{x}_p - \boldsymbol{m}_i^{(t)}\|^2 \leq \|\boldsymbol{x}_p - \boldsymbol{m}_j^{(t)}\|^2, \;\; \forall_j, \; 1 \leq j \leq k\}. \tag{10}$$

During the update step each cluster centers location is updated by taking the mean of the observations assigned to that particular cluster center,

$$m_i^{t+1} = \frac{1}{|S_i^{(t)}|} \sum_{\boldsymbol{x}_j \in S_i^t} \boldsymbol{x}_j. \tag{11}$$

This iteration is continued until convergence which is defined as when the assignment step stays the same. This can be proven to happen in a finite number of steps but it is not guaranteed to be a global optima. To counter this the k-means is usually run multiple times with different starting centers and the best choice is selected.

One of the main issues is selecting the number of cluster centers, k. This number can be selected and analyzed in multiple ways but it is not something that is discussed in this thesis. In our case we are limited by our computational power and will thus select the maximum number of clusters.

3.5 Image features

An image contains a massive amount of data. In order to run classification algorithms in a reasonable amount of time we need to reduce this data. The goal of the following algorithms is to reduce an image into a rather small feature vector, also called image descriptor. This descriptor should some sense capture the most important characteristics of the images, for example the colors or the textures.

3.5.1 Color histograms

Color is the most obvious way to describe a scene without explicitly treating the objects. Similar scenes should have similar colors, for example playing football should result in multiple green images while sailing should mostly consist of blue images. A pixel has three dimension where each take 256 different value, this results in over 16 million different colors. This is too many dimensions to describe an image if we also want the calculations to be feasible. By taking a 64-bin histogram in each dimension and concatenate these we get a 192-feature vector.

We do this using the three different color space models. Thus resulting in three different feature vectors:

- 192-bin RGB histogram feature vector
- 192-bin HSV histogram feature vector
- 192-bin LAB histogram feature vector

3.5.2 ECOH

ECOH is short for edge and color orientation histogram and is a holistic feature that tries to capture the distribution of lines and color in an image. The basic idea is that a certain type of scene should have a typical distribution of edges and color, for example a city image should contain more vertical lines than a more natural environment. The same should be true for colors as described in the previous section.

ECOH was introduced by Kim et al. [6], as a modification and extension to EOH [8] applied to indoor-outdoor segmentation. The basic ECOH theory could be implemented in various ways but in this thesis we will use the original implementation as follows:

We divide the image into blocks as the scheme described in figure 6. We will compute the ECOH for each block separately and then concatenate the blocks into one full length feature. During the concatenation we introduce different weights for the blocks in order to focus on the areas of interest, in particular more emphasized will be placed on block one, two and five. The original reason in [6] for this scheme was that humans are usually in the middle of a photo. It doesn't matter if the image was taken inside or outside, if a human is in the image it is likely they are in the middle. Thus, focus are concentrated on areas that carries information about the scene.

The same argument cannot be made by images taken by the Narrative Clip since no active choice was made while taking the image. To really understand and improve this process we should test multiple block formations and different weights. This will not be done due to the time limitations described in the section 1.3, instead we will make the following argument:

Narrative Clip is normally worn in a way such that the images are taken at shoulder height and pointing forward. Due to Narrative Clip's accelerometer all images taken will have a horizontal horizon. This mean that a city scene will have a higher likelihood for sky in block one, buildings in block two and five and possibly a street in block three and four. With the same

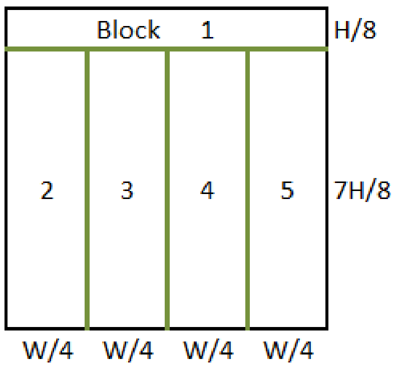

Figure 6: An image divided into blocks of a specific size.

reasoning and numbering, an indoor scene might consist of roof, walls and an object. Skimming through the images confirms the importance of block 1, while the distinction between the border blocks (2 and 5) and the center blocks (3 and 4) is not particular obvious. That is why we choose the following weights for the different sections:

$$w_1 = 1, w_2 = 0.5, w_3 = 0.5, w_4 = 0.5, w_5 = 0.5 \tag{12}$$

First the image, I, is convolved separately with a horizontal Sobel filter

$$G_x(x, y) = Sobel_x * I(x, y) \tag{13}$$

and a vertical Sobel filter

$$G_y(x, y) = Sobel_y * I(x, y). \tag{14}$$

A sobel filter detects edges in a certain direction, in this case the x and y direction. By combining these we are able to detect the edge strength independent of the direction.

$$G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \tag{15}$$

This enables us to remove noise and other edges that are not strong enough by simple thresholding,

$$G(x, y) = \begin{cases} G(x, y) & G(x, y) \geq T \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The orientation of the line can be found by calculating the angle. We don't make any difference between opposite angles, that is any angle between π and 2π is mapped to the corresponding angle between 0 and π. This is done because the same edge can have a different sign depending on the lightning, shining from one direction or the other should not affect the edge detection.

$$\theta(x,y) = mod(arctan2(G_y(x,y), G_x(x,y)), \pi) \tag{17}$$

In order to better handle the data we limit the number of orientations to K_E and only save the edge strength for these orientations

$$\phi_k(x,y) = \begin{cases} G(x,y) & \pi\frac{k}{K_E} < \theta(x,y) \leq \pi\frac{k+1}{K_E} \\ 0 & \text{otherwise} \end{cases} \quad where\ 0 \leq k < K_E \tag{18}$$

For each block we now want to calculate the amount of edge strength in each direction. That is, we want to create a histogram for each block where the bins are the k orientations and the height the cumulative edge strength.

$$E_{ik} = \sum_{(x,y)\in Block_i} \phi_k(x,y), \quad 1 \leq i \leq 5 \tag{19}$$

Since all the $\psi_k(x,y)$ are positive we can speed up these calculations using integral images introduced in [2].

Thus far we have only created the edge oriented histogram for each block. We are now interested in doing a similar transformation for the colors, the HSV colorspace already provides an appropriate coordinate system as described in section 3.2.2. The hue, H, is equivalent to the angle and the saturation, S, is equivalent to the strength. The only difference is that we use the complete rotation from 0 to 2π for the hue channel since the angle represent different colors. We limit the number of colors by segmenting H into K_C different orientations. The corresponding color oriented histogram is thus described by

$$\gamma_k(x,y) = \begin{cases} S(x,y) & 2\pi\frac{k}{K_C} < H(x,y) \leq 2\pi\frac{k+1}{K_C} \\ 0 & \text{otherwise} \end{cases} \quad where\ 0 \leq k < K_C. \tag{20}$$

Which we collect for each block,

$$C_{ik} = \sum_{(x,y)\in Block_i} \gamma_k(x,y), \quad 1 \leq i \leq 5. \tag{21}$$

Finally we concatenate all the histograms from all the blocks using different weights for each block. We chose $K_E = K_C = 8$ and concatenated the results according to this scheme

$$\begin{aligned} \mathbf{F} &= (w_1\mathbf{f}_1, w_2\mathbf{f}_2, w_3\mathbf{f}_3, w_4\mathbf{f}_4, w_5\mathbf{f}_5), \\ \mathbf{f}_i &= (E_{i1}, C_{i1}, E_{i2}, C_{i2}, ..., E_{iK}, C_{iK}). \end{aligned} \tag{22}$$

3.5.3 GIST

Oliva and Torralba [14] approached scene analysis by looking at how humans characterize a scene. They found that concepts like naturalness, openness, roughness, expansion and ruggedness played a vital roll in humans' scene understanding. They explained it as capturing the gist of the scene. These concepts are quite abstract, for example man-made objects tend to have straighter lines than natural objects.

Oliva and Torralba used spectral information to estimate and train an algorithm to distinguish the different characteristics. A more naive gist descriptor was proposed in [13] and is what we have used in this thesis. This approach uses a set of gabor filters [12], visualized in figure 7, to filter an image into a number of filter responses. The image is then divided into patches each being 4×4 pixels, for each patch the mean filter response is calculated for each filter response. Various structures will be captured by each filter depending on the size and rotation, for example a vertical filter will capture buildings better than an angled filter. The Gabor filter was chosen specifically for its similarities with cortical cells responsible for the human visual system [11]. The hope is that these filters will resemble the spectral approach presented in the original gist implementation.

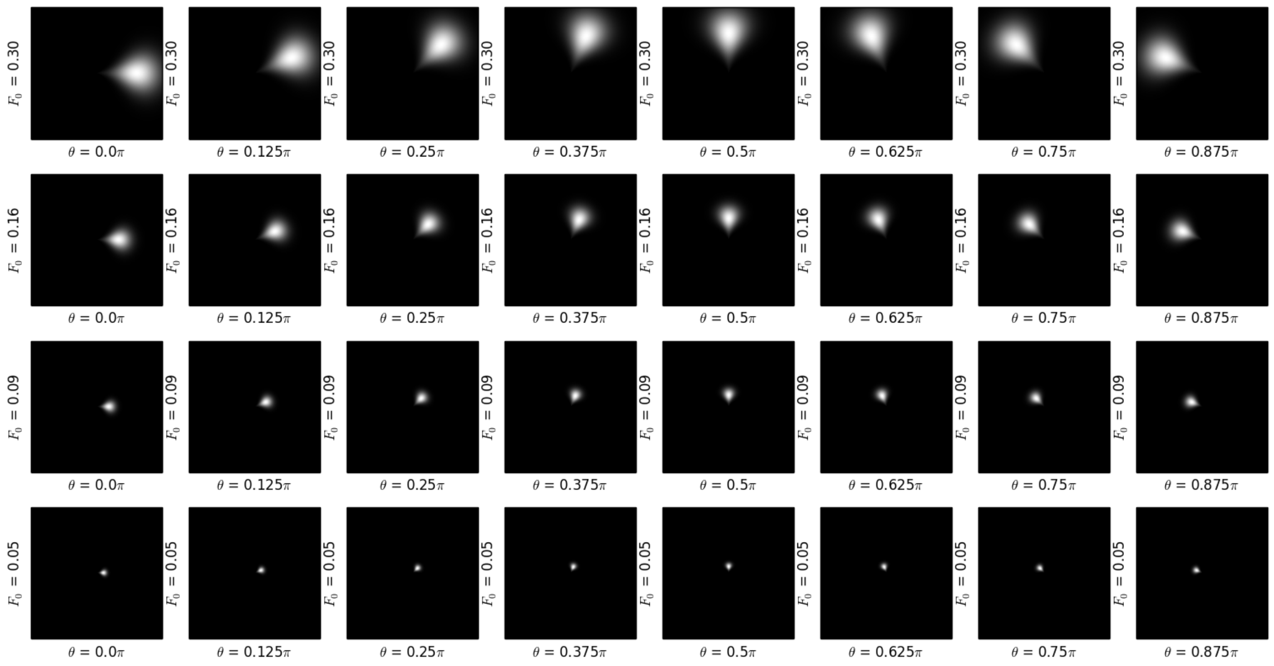

Figure 7: The power spectrum of the different filters used in the GIST implementation. The 32 filters are all Gabor filters but at four different sizes and eight different angles.

3.5.4 Centrist

Centrist features aim to find a distribution of smaller structures that together form a description of a larger scene. The idea is that a scene have a higher likelihood of certain structures than other scenes. For example some scenes might have more circle or rectangle shaped structures than other scenes. The centrist algorithm does not explicitly look for structures like circles or rectangles. Instead it takes the *census transform* of the image and calculate a histogram. Circles tend to generate certain histogram structures and the goal is to capture these different local structures. Together should these structures generate a histogram that is characteristic for a particular scene.

Census Transform Census Transform is a transformation that uses its local surroundings to calculate a pixels value. It is done in a non-parametric way that prevents the use of convolution. Instead it is calculated in the following manner.

Starting from the top-left corner of a three by three pixels larger patch the neighboring pixel is compared with the middle pixel. If the middle is greater than or equal to the neighbor a one is saved, otherwise a zero is saved instead. This is done for each of the eight pixels around the middle pixel going left to right and top to bottom. In the process an eight bit large array is created. The array can be seen as a binary number between 00000000 and 11111111. The first bit is considered the most significant, that is the upper left corner. This binary number be converted into a decimal number ranging between 0 and 255. This value is saved in a new matrix at the same location as the middle pixel. See 8 for an example.

$$\begin{array}{c|c|c} 10 & 40 & 90 \\ \hline 50 & 50 & 90 \\ \hline 70 & 20 & 80 \end{array} \Rightarrow \begin{array}{ccc} \textcircled{1} & 1 & 0 \\ 1 & & 0 \\ 0 & 1 & 0 \end{array} \quad \begin{array}{c} \text{bit5} \\ \Lsh \\ \end{array} \Rightarrow (1101\mathbf{0}010)_2 \Rightarrow 205$$

Figure 8: An example of how to calculate the census transform for a single pixel, in this case the middle pixel, 50.

Doing this transformation for all pixels in the image will generate a new image where the global structures are intact but the local details like the textures are suppressed. The key to this behavior is constraints that are applied during the census transform. The main constraint is that each pixel have a complement pixel in another patch. Consider the following transformation, which is one step to the right from example 8.

Bit 5 in figure 8 is the middle pixel in figure 9 and bit 4 in figure 9 is the middle pixel in figure 8. Bit 4 and 5 is connected between different patches and unless they have the same pixel value the two bits are opposite

$$\begin{array}{c|c|c} 40 & 90 & 120 \\ \hline 50 & 90 & 20 \\ \hline 20 & 80 & 100 \end{array} \Rightarrow \begin{array}{cccc} \text{bit4} & 1 & 1 & 0 \\ \Lsh & 1 & & 1 \\ & 1 & 1 & 0 \end{array} \Rightarrow (110\mathbf{1}1110)_2 \Rightarrow 222$$

Figure 9: Similar example as figure 8 except one pixel to the right. This shows the relationship between the bit 4 here and bit 5 in figure 8. The bits are bold to show that they are the opposite of each other.

of each other. There are other constraints like these as well, some are less direct by taking paths through multiple pixels. The concept is that the pixels are constrained by each other and taking the histogram an census transformed image captures more than a simple gray-level distribution. In [22] it was shown that the histogram also carries parts of the global structure, a shuffled image could be reconstructed to a certain extent just by trying to match the histograms.

The histogram is calculated using 254 different pixel values as bins. We remove the complete white and black pixels since they are too common due to how the census transform is calculated. As such they do not provide much information. Due to the constraints discussed above multiple bins in the histogram are correlated thus making a principal component analysis suitable in order to limit the amount of dimensions. In [22] the number of dimensions were reduced to 40, we used the same in this thesis.

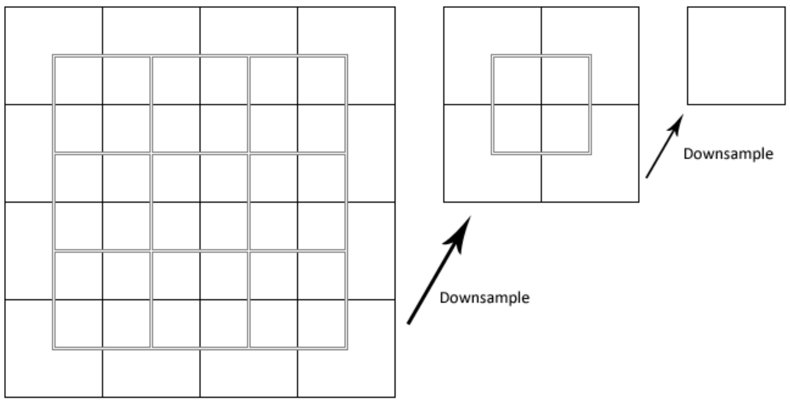

Figure 10: Illustration over how the spatial pyramid segments are divide and downsampled.

In [22] a spatial pyramid was presented, it works by first dividing the original image into 25 overlapping segments as seen in 10. The image is down-sampled to 1/4th of the original size and another 5 segments are created. Finally the image is once again down-sampled to 1/16th of the original size and one single segment is created. The purpose for this scheme is to increase sensitivity to different sizes and frequency components. By using this scheme we find structures at different sizes that could increase the performance in scene recognition. For example, a scene looks different depending on how far away the image was taken.

For the centrist descriptor we calculate the census transform for each level and compute the histogram for each different segment. We reduce the dimensions of each by using a unique PCA for each segment. Finally we concatenate all the histograms into one feature vector, resulting in a vector containing $40 * (25 + 5 + 1) = 1240$ elements.

3.5.5 SIFT

Objects are vital for scene recognition and especially how the relate to each other. For example recognizing a boat is almost a guarantee that the image is taken close to water, recognizing multiple boats and the image is likely taken close to a marina. The main issue with objects is the vast amount of objects that exists, especially at different scales. A human can be seen as one object but also as an object made up out of different body parts. Large amount of data is needed to recognize even the basic objects and is thus outside the scope of this thesis. Instead we will use a naive approach in order to describe a scene using objects. The idea is that similar scenes should contain similar objects, the most prominent features of these objects should also be similar. Images depicting the same scene should thus have a similar distribution of these prominent features and could thus be a good scene descriptor. Using this approach avoids detecting any object explicitly and thus removing the necessity of training the algorithm for each object. Instead we train the data on the distribution of the most relevant points and hope to capture the underlying objects.

An object can be viewed from different distances, at different rotations and at different illuminations. We are interested in detecting certain characteristic points on the object and thus need a method that detect these points independent of how the object is viewed. Scale-invariant-feature-transform (SIFT) is one method for this task and is also fairly robust to the various changes, in particular to scaling and rotating the object. SIFT was developed by David Lowe and has been a popular local-feature detector since it was introduced 1999 [9].

The SIFT-algorithm The first stage of the SIFT-algorithm is to detect the interesting points, called keypoints. *Laplacian of Gaussian* (LoG) has a strong response to blobs, that is, areas that have similar range of values. The blobs comes at multiple different scales and we need to calculate the LoG for each. The LoG is fairly expensive to calculate and Lowe introduced an approximation in order to reduce the runtime. The *Gaussian* (G) of an image at different scales is defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{23}$$

The LoG is defined as

$$L(x, y, \sigma) = \frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G. \tag{24}$$

The left-hand side can be approximated as

$$\frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}, \tag{25}$$

which can be rewritten as

$$D(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G. \tag{26}$$

The left hand side is a *Difference of Gaussian* (DoG) and is an approximation to the LoG. Thus by calculating multiple Gaussians and taking the difference between two adjacent scales we are able to approximate the LoG at certain scales. We are interested in finding points where we have a strong local response to the DoG, that is where we either have a local maximum or minimum, a so called extremum. These points should be local extreme points not only at the current scale but also looking at the two adjacent scales as visualized by figure 11.

By doing this for all the different pixels and scales we get numerous keypoint candidates. In order to improve the stability of the local extreme points a Taylor expansion of $D(x, y, \sigma)$ is used. The derivative of this function determines the location of the extremum. A thresholding of the value at these points removes any low-contrast keypoints and thus increases the invariance towards illumination changes. Lowe proposed that the keypoint should have a $|D(\hat{x})| > 0.03$ which is what has been used in this thesis.

The DoG response well to blobs and corner but also edges, an edge is not a good local feature since it can be shifted along the edge and still give an almost identical response. In order to remove these points an Hessian matrix is created

$$\boldsymbol{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{27}$$

It can be shown that

$$\frac{Tr(\boldsymbol{H})^2}{Det(\boldsymbol{H})} = \frac{(r+1)^2}{r}, \tag{28}$$

where r is the ratio between the two eigenvalues of $\boldsymbol{H}$. An edge has a large ratio between the two eigenvalue while a corner has similar eigenvalues.

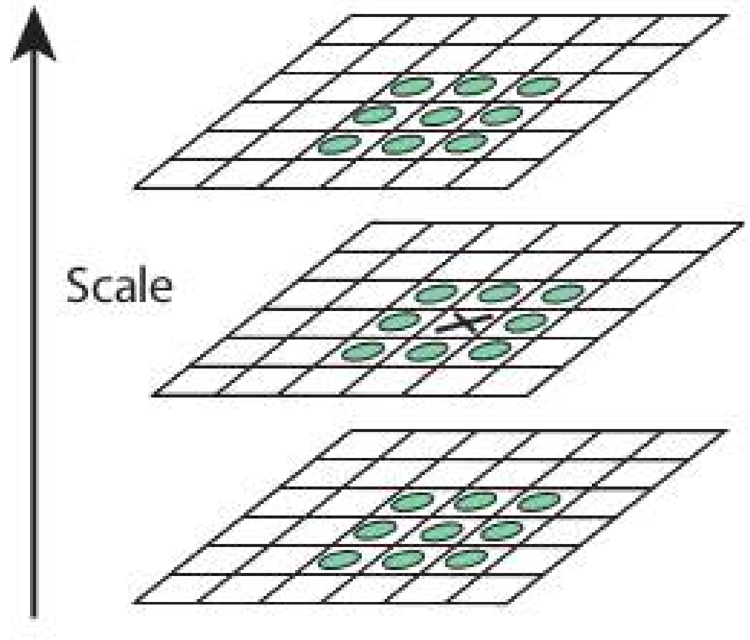

Figure 11: Illustration over which pixels we check to see if the pixel, marked with an X, is a local extreme point.

Lowe proposed to remove all keypoints where $r > 10$ which is what has been used in this thesis.

We have now removed all the questionable keypoints that either lacked contrast or stability. In order to achieve rotation invariance we now assign an orientation for each keypoint, the remaining task of creating a descriptor will use this orientation as a base in order to ensure rotation invariance. The orientation assessment is done in a Gaussian-weighted circular window with a σ 1.5 times larger than the current keypoints scale. In this window we calculate the magnitude and orientation for each location.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{29}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{30}$$

The orientations are binned to 36 different bins and we simply add the magnitude to each bin depending on the orientation. This is done for all the pixels and we get a histogram for each keypoint. The orientation with the largest magnitude is chosen as a keypoint. There might be other large peaks as well, if any of these are more than 80% of the maximum peak we create keypoints for these as well. Thus it is possible for one keypoint location to generate one or more keypoints. Although it doesn't happen in most cases.

Finally we construct the actual local feature descriptor. This is done by breaking down a 16×16 pixels larger window around the keypoint into 4×4 blocks. As before we calculate the magnitude and orientation for each pixel. We rotate these results using the keypoints orientation, this is done to ensure rotation invariance. For each block 4×4 we calculate an orientation histogram with 8 bins. The difference here compared to previously is that we add the gaussian window adjusted magnitude for each pixel to the corresponding orientation bin. This limit errors that is more common to arise further away from the keypoint. In total we get 16 blocks with 8 orientations each thus resulting in a 128-dimensional local feature vector.

Creating an image descriptor The SIFT algorithm generates multiple local feature vectors. They are highly discriminative and can generate an unknown number of features. Although the discriminative nature makes it possible to distinguish between two different but similar keypoints there should still be certain similarities. Considering an object is made up by multiple keypoints it should be possible to capture this correlation. Therefor we apply the K-means clustering method as described in section 3.4. The goal is to capture the most common local feature vectors, these hopefully make up important aspects of an image that could be suitable for scene recognition.

The K-means algorithm generates a number of cluster centers, all the local feature vectors are assigned to the closest cluster center. We simply count how many of each cluster center exists in a new image and thus creating a histogram of the local feature vectors. After dividing by the number of vectors we get an image descriptor suitable for detecting object distribution changes in our images. The length of the descriptor is the same as the number of cluster centers. As stated in section 3.4 we select the largest computational feasible number of cluster centers, in this case that is 95 cluster centers.

3.5.6 Textons

Textures have a vital role in scene understanding, giving us a way to distinguish the sky from the ground if the color channel is missing. Textures comes in a wide range of shapes and sizes making it hard to define a particular texture mathematically. For example, a chair is normally not a texture but if we view it at a distance it could be viewed as one. The chair is a rather complex texture that can be broken down into simpler parts. The simplest textures are bars and blobs at different sizes and rotations, these textures are the basic building blocks of textons [7].

By filtering an image with a wide range of different filters we get a unique response for a particular texture. Similar textures, like grass at different locations, will have a similar response to the filters. In this thesis we used the 48 filters shown in figure 12 but multiple other filters could also be used. The filters used are:

First derivative of Gaussian: A total of 18 filters at six different orientations, $\theta = (\frac{\pi}{6}, \frac{2\pi}{6}, \frac{3\pi}{6}, \frac{4\pi}{6}, \frac{5\pi}{6}, \pi)$ and at three different scales with an elongation factor of 3, $\sigma_x = (\sqrt{2}, 2, 2\sqrt{2})$ and $\sigma_y = 3\sigma_x$.

Second derivative of Gaussian: A total of 18 filters at six different orientations, $\theta = (\frac{\pi}{6}, \frac{2\pi}{6}, \frac{3\pi}{6}, \frac{4\pi}{6}, \frac{5\pi}{6}, \pi)$ and at three different scales with an elongation factor of 3, $\sigma_x = (\sqrt{2}, 2, 2\sqrt{2})$ and $\sigma_y = 3\sigma_x$.

Laplacian of Gaussian: Eight filters with different scales, $\sigma = (\sqrt{2}, 2, 2\sqrt{2}, 4, 3\sqrt{2}, 6, 6\sqrt{2}, 12)$.

Gaussian: Four filters with different scales, $\sigma = (\sqrt{2}, 2, 2\sqrt{2}, 4.)$

16 first derivative of Gaussian filters, 6 with

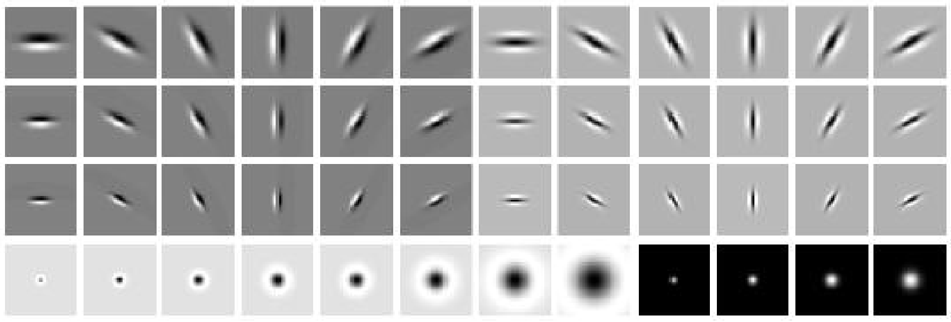

Figure 12: The 48 different filters used to create the different textons [7].
Row 1-3, column 1-6: Derivative of Gaussian filters
Row 1-3, column 7-12: Second derivative of Gaussian filters
Row 4, column 1-8: Laplacian of Gaussian filters
Row 4, column 9-12: Gaussian filters

If we apply the filters to multiple images at all the different pixel locations we get a large set of filter responses. We use this set to find multiple cluster centers using a k-means algorithm as described in section 3.4. The goal is to gather similar textures into one cluster, for example should all grass textures be in the same cluster. Since a cluster center describes the same texture it has in literature been defined as a texton. In other words the textons are the most essential textures of the training set.

For a new image we calculate the filter responses for all the pixels. Each filter response are then compared with the textons, the closest texton gets assigned to the specific pixel. This generates a new image consisting only of the texton designations. The new image has a smaller filesize but the primarily benefit is that each pixel is more meaningful since it describes a texture not only a gray value. Similar to the SIFT, we collect the number of each textons and divide by the number of pixels. This histogram will be a suitable image descriptor to detect texture changes in our images. The size of the descriptor is equal to the number of textons or cluster centers. Like the SIFT-feature we chose 95 cluster center simply due to computational limitations.

3.6 Classification methods

Classification methods are used to classify data in a supervised setting. The goal is to construct algorithms that take a new observation and classify it correctly. The algorithms need a set of annotated observations to adjust or train the parameters in order to be able to classify a new observation correctly.

3.6.1 Support Vector Machine

Support Vector Machine [1] is a supervised non-probabilistic machine learning method used to label data into two different classes. It takes a supervised training set, consisting of pairs of labels and feature vectors, and constructs a hyperplane that separates the two classes. To classify a new feature vector the model simply check at which side of the hyperplane the feature can be found and classify it accordingly.

Imagine a linear separable training set consisting of two-dimensional feature vectors as depicted in figure 13. The goal of the SVM is to find the line that best separates the two classes, that is, the distance from the line to the closest feature vector should be as large as possible for both classes. This is equivalent to saying that the distance from the line to the closest feature vector is the same for both classes. These vectors that define this distance are called the support vectors.

Given a linear separable training set

$$H = \{(\boldsymbol{x_i}, y_i) \mid \boldsymbol{x_i} \in \mathbb{R}^p,\ y_i \in \{-1, 1\}\}_{i=1}^{n} \tag{31}$$

we can construct a hyperplane, $(\boldsymbol{w}, b)$, such that any point $\boldsymbol{x}$ on the hyperplane will satisfy

$$\boldsymbol{w} \cdot \boldsymbol{x} - b = 0. \tag{32}$$

In order for the hyperplane to separate the training set we need to constrain it. In other words we need to define $\mathbf{w}$ and b. Since we know the data is separable one way would be to force all the positive samples to be larger or equal to one and all the negative samples to be less than or equal to

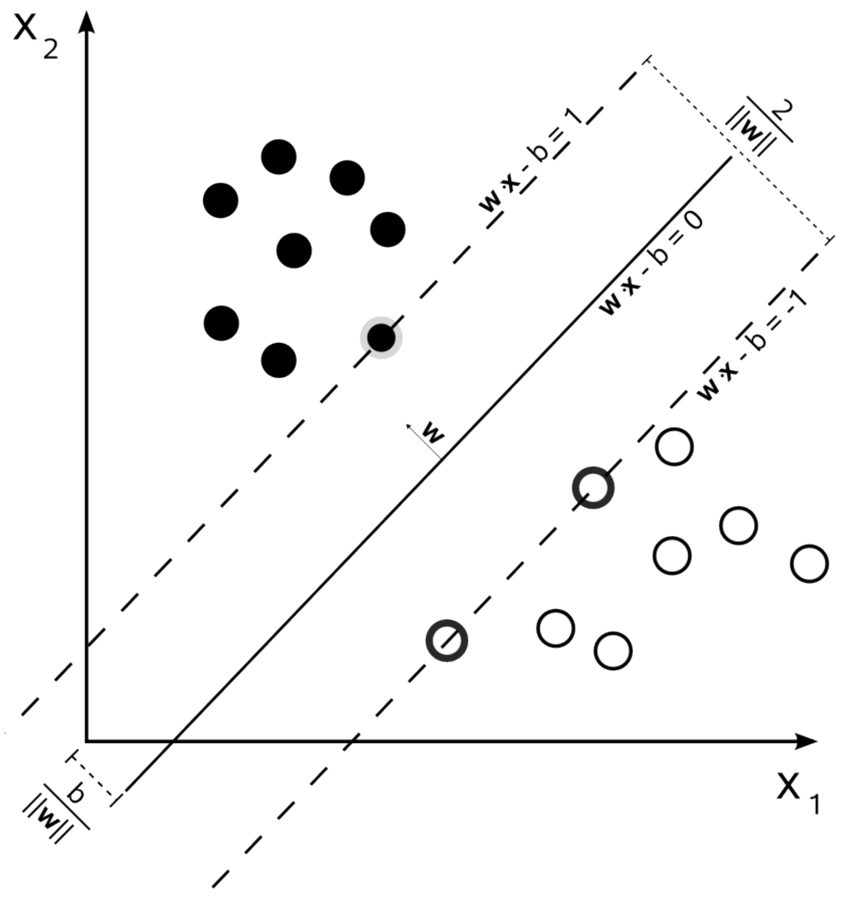

Figure 13: Example of a SVM classification for two classes. The black dot belong to one class and the white dots to another. The filled line is the optimal hyperplane.

negative one. It is out of mathematical convenience we choose equal to one, the main point is that different samples should have different signs. This is mathematical equivalent to

$$y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1, \quad \forall i \tag{33}$$

which creates two new hyperplanes as shown in Figure 13.

It can be shown that the distance between the two constraining hyperplanes is equal to $2/\left\|\boldsymbol{w}\right\|$ and thus the problem is equivalent to finding

$$\begin{gathered} \underset{w,b}{min} \; \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 \\ subject\ to: \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1, \quad \forall i. \end{gathered} \tag{34}$$

Equation 34 is not, in its current form, trivial to solve due to the quite complex constraints. We have two unknowns inside a linear constraint and we want to minimize a convex function. This limits where the minimum can be found, it either has to be located at the global minimum or somewhere along the constraint. This class of problems are called *quadratic programming optimization problem* and can be solved using appropriate techniques. For these kind of problem we can usually simplify the quadratic programming by reformulate the problem using Lagrange multipliers

$$\begin{gathered} min\ L_p \equiv \frac{1}{2}\left\|\boldsymbol{w}\right\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\boldsymbol{w}\cdot\boldsymbol{x_i}+b)-1] \\ subject\ to: \quad \alpha_i \geq 0, \quad \forall i. \end{gathered} \tag{35}$$

This technique removes the complicated constraints and replace them with a far more simple one; the Lagrange multipliers α_i must be positive. The price for this operation is a more complicated minimization function. A fairly small price considering that a minimum requires the derivative to be equal to zero,

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i y_i \boldsymbol{x_i} = 0 \tag{36}$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{37}$$

Using the results from 36 and 37 in 35 and reformulating into a maximization problem,

$$\begin{gathered} max\ L_d = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x_i}\cdot\boldsymbol{x_j} \\ subject\ to: \quad \sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad \forall_i \end{gathered} \tag{38}$$

It is still a quadratic programming problem but a much easier one to since we only have rather simple constraint left. This problem can be solved by standard quadratic programming solvers that will not be covered here. The solver will return the different Lagrange coefficients which can be used to find the hyperplane parameters. $\boldsymbol{w}$ is found using equation 36 and b is found using equation 32 and the support vectors that define the hyperplane.

In most cases the data is not fully separable. To handle this we introduce an error on each observation that depends on the distance to the hyperplane. The derivation is similar as before and will not be shown here, instead we refer to [1] for a detailed derivation.

Furthermore, the data might not be linearly separable. It is possible the data is better separated by a polynomial function or perhaps radial function. In equation 38 the term $\boldsymbol{x_i}\cdot\boldsymbol{x_j}$ appeared, this term is called a linear kernel and belongs to a family called Kernel Functions. It defines the behavior of the hyperplane, in this case a linear function. By replacing this term with

$k(\boldsymbol{x_i}, \boldsymbol{x_j})$ we can use different kernels that live in a higher dimensional space. Structures that previously weren't separable by a plane might now be and we can use this to get a better classification. Without going into to any detail, one of the most common kernels is the radial basis kernel,

$$k(\boldsymbol{x_i} \cdot \boldsymbol{x_j}) = e^{-\frac{\left\|\boldsymbol{x_i}-\boldsymbol{x_j}\right\|^2}{2\sigma^2}}. \tag{39}$$

In this thesis this kernel will be used exclusively. Time-constraints restricted proper comparison against the linear kernel and the radial basis function for no particular reason. The complete derivation using kernels will not be performed here, for a complete derivation see [1].

Given an observation the SVM classification generates the distance to the hyperplane. Depending on the sign it classifies the observation as one of the two classes. In some cases it is more useful to have the probability that an observation belong to a class rather than a distance. In [17] Platt fitted a logistic regression to the distances generated by the training set in order to get a probabilistic-SVM. This modification makes the SVM able to return the probability that an observation belong to a certain class. We will not explain this method further and refer to [17] for a full explanation.

3.6.2 Hidden Markov Models

Hidden Markov Model (HMM) [18] is a way to determine the class or state of multiple time consecutive observations. Consider M consecutive observations

$$\boldsymbol{O} = \{\boldsymbol{O_1}, \boldsymbol{O_2}, ..., \boldsymbol{O_M}\} \quad where\ \boldsymbol{O_k} \in \mathbb{R} \tag{40}$$

and the following N states

$$\boldsymbol{S} = \{S_1, S_2, ..., S_N\}. \tag{41}$$

where each observation belongs to a certain state that is unknown or hidden

$$\boldsymbol{Q} = \{q_1, q_2, .., q_M\} \quad where\ q_k \in \boldsymbol{S}. \tag{42}$$

We are interested in finding this unknown state sequence $\boldsymbol{Q}$ given an observation $\boldsymbol{O}$, a problem well suited for a *Hidden Markov Model.* Since the observations have a certain degree of noise the problem turns into finding the optimal state sequence given the observation, in this context optimal means most likely.

A new observation usually depends on the previous one, for our usage this means that if the previous images was taken outside it is likely that the next image will be as well. The Markov property in the HMM takes this further, instead of looking at all the previous observations only the latest observation's state is taken into account.

In order to calculate the HMM we will need the following variables. The state transition probability distribution

$$a_{ij} = P[q_{t+1} = S_j | q_k = S_i], \quad 1 \leq i, j, \leq N. \tag{43}$$

This property describes how likely certain states are to transition into others, for example it might be more likely that the next day after a sunny also is sunny rather than rainy.

The observation probability distribution for each state

$$b_j(k) = P[\boldsymbol{O}_k | q_k = S_j], \quad 1 \leq j \leq N, \quad 1 \leq k \leq M. \tag{44}$$

Certain observations are more probable for some states than others. If we see the sun in the morning it might be more likely that the day will be sunny rather than rainy.

The initial state distribution

$$\pi_j = P[q_1 = S_j], \quad 1 \leq j \leq N. \tag{45}$$

Each sequence has a starting state, π describes the likelihood for each. All these variables can be group as $\lambda = (a, b, \pi)$. λ is essential as it defines all the required properties that needs to be configured in order to have a properly working HMM. λ can either be chosen or, more common, trained using multiple training sequences.

Consider the probability that a certain state sequence $\boldsymbol{Q}$ generates the observation $\boldsymbol{O}$. This can be written as

$$P(\boldsymbol{O}|\boldsymbol{Q}, \lambda) = \prod_{k=1}^{M} P(\boldsymbol{O}_k | q_t, \lambda) = b_{q_1}(\boldsymbol{O_1}) \cdot b_{q_2}(\boldsymbol{O}_2) \cdots b_{q_T}(\boldsymbol{O}_T) \tag{46}$$

The probability for $\boldsymbol{Q}$ is

$$P(\boldsymbol{Q}|\lambda) = \pi_{q_i} a_{q_1 q_2} a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \tag{47}$$

This allow us to calculate the joint probability distribution using conditional probability.

$$P(\boldsymbol{O}, \boldsymbol{Q}|\lambda) = P(\boldsymbol{O}|\boldsymbol{Q}, \lambda) \cdot P(\boldsymbol{Q}|\lambda) \tag{48}$$

These calculations are completely impractical for anything but the smallest datasets. The problem with this approach is that for each state the algorithm needs to calculate all the possible state changes. Most of these changes are the same for all the states thus resulting in multiple unnecessary calculations. This can be solved by using a dynamic programming approach that solves the problem by saving each step and thus eliminating unnecessary calculations. This is possible due to the Markov property, when calculating the probability at time t we are only interested in the N different states at time $t-1$, any previous knowledge is irrelevant. In fact we can use the dynamic programming in two ways, calculating from the beginning or from the end. The forward variable is defined as

$$\alpha_t(i) = P(\boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O_t}, q_t = S_i|\lambda) \tag{49}$$

and can be calculated as follows,

$$\alpha_1(i) = \pi_i b_i(\boldsymbol{O_1}), \quad 1 \leq i \leq N \tag{50}$$

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right) b_j(\boldsymbol{O}_{t+1}), \quad 1 \leq t \leq T-1, \quad 1 \leq j \leq N. \tag{51}$$

The backward variable is defined slightly differently

$$\beta_t(i) = P(\boldsymbol{O}_{t+1}, \boldsymbol{O}_{t+2}, \cdots, \boldsymbol{O}_T | q_t = S_i, \lambda) \tag{52}$$

and can be calculated in a similar but slightly different manner,

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \tag{53}$$

$$\beta_t(i) = \left(\sum_{j=1}^{N}\right) a_{ij} b_j(\boldsymbol{O}_{t+1}) \beta_{t+1}(j), \quad T-1 \geq t \geq 1, \quad 1 \leq i \leq N. \tag{54}$$

We are interested in finding the most likely state sequence $\boldsymbol{Q}$ given an observation and parameter λ. We want to calculate this in a dynamic programming manner and thus the following is of interest

$$\gamma_t(i) = P(q_t = S_i|\boldsymbol{O}, \lambda) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)} \tag{55}$$

The most likely state for a given time, t, can then be found as,

$$q_t = \underset{1 \leq i \leq N}{argmax}\big[\gamma_t(i)\big], \quad 1 \leq t \leq T. \tag{56}$$

The Viterbi algorithm The problem with the approach in 56 is that it calculates each step independently. In certain situations we might have state transitions with zero probability which can't be handled by equation 56. Instead we try to find the "single best state sequence", that is, maximizing $P(\boldsymbol{Q}|\boldsymbol{O}, \lambda)$ or equivalently maximizing $P(\boldsymbol{Q}, \boldsymbol{O}|\lambda)$. This is done by the Viterbi algorithm and starts by defining

$$\delta_t(i) = \underset{q_1, q_2, \cdots q_{t-1}}{max} \Big[P\big(q_1, q_2, \cdots, q_t = i, \boldsymbol{O}_1, \boldsymbol{O}_2, \cdots, \boldsymbol{O}_t|\lambda\big)\Big], \tag{57}$$

which can be reformulated using induction

$$\delta_{t+1}(j) = \underset{i}{max}\big[d_t(i) a_{ij})\big] \cdot b_j(\boldsymbol{O}_{t+1}). \tag{58}$$

$\delta_t(i)$ describes the highest probability to generate the state i at time t. It takes into account all the previous path but only saves the optimal one. In other words equation 58 is used to calculate the best state sequence and is implemented and used

$$\delta_1(i) = \pi_i b_i(\boldsymbol{O_1}), \quad 1 \leq i \leq N \tag{59}$$

$$\psi_1(i) = 0 \tag{60}$$

The ψ variable is used to store the best state sequence when we use recursion to calculate the different states

$$\delta_t(j) = \underset{1 \leq i \leq N}{max}\big[\delta_{t-1}(i)a_{ij}\big] \cdot b_j(\boldsymbol{O}_t), \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{61}$$

$$\psi_t(j) = \underset{1 \leq i \leq N}{argmax}\big[\delta_{t-1}(i)a_{ij}\big], \quad 2 \leq t \leq T, \quad 1 \leq j \leq N \tag{62}$$

The final step saves the results separately

$$P^* = \underset{1 \leq i \leq N}{max}\big[\delta_T(i)\big], \tag{63}$$

$$q_T^* = \underset{1 \leq i \leq N}{argmax}\big[\delta_T(i)\big]. \tag{64}$$

Finally we backtrack to find the optimal state sequence

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad T-1 \geq t \geq 1, \tag{65}$$

where q_t^* is the final result and gives us the single best state sequence given a certain observation.

3.7 Changepoint detection

Changepoint detection is used to detect distribution changes in a sequence. Various changes can be detected depending on the method, it could be a change in distribution type or perhaps the same distribution but with parameter variations, for example a change in mean. In this thesis the interest is limited to the last one, we want to detect changes in the mean value. This will be done by using a distribution independent method based on *cumulative sum control chart* (CUSUM) first presented by Page [15]. This method have been improved since, in particular by Hinkley and Schechtman [5] who proposed a bootstrap method to detect a change and Taylor [21] who proposed a method to detect multiple changes in an iterative way. In this thesis we extend Taylor's method to work at multiple dimensions and thus detect multiple mean changes on multivariate data.

The CUSUM method takes the cumulative sum of the difference between the mean and the datapoints. Consider a sequence $\boldsymbol{X} = X_1, X_2, ..., X_N$ with mean

$$\bar{X} = \frac{X_1 + X_2 + ... + X_N}{N} \tag{66}$$

The CUSUM method is defined as

$$\begin{aligned} S_0 &= 0 \\ S_i &= S_{i-1} + (X_i - \bar{X}) \end{aligned} \tag{67}$$

The CUSUM will always sum to zero since we are subtracting the mean N times. Thus what the chart is actually saying is how far away from the mean we are given a sequence number. Now consider we have two different means with the following relation,

$$\bar{X}_1 = \frac{X_1 + X_2 + ... + X_m}{m} > \frac{X_{m+1} + X_{m+2} + ... + X_N}{N - m} = \bar{X}_2 \tag{68}$$

Thus the mean, $\bar{X}$, is in the middle between $\bar{X}_1$ and $\bar{X}_2$,

$$\bar{X} = \frac{\bar{X}_1 + \bar{X}_2}{2}. \tag{69}$$

This mean that any X_i up to the point $m + 1$ probably is larger than $\bar{X}$ and thus having a positive contribution to S. While any point after m is probably smaller than $\bar{X}$ and thus having a negative contribution. This mean that S will increase in the beginning up to the point m after which it will decrease. If we instead had a situation where $\bar{X}_1 < \bar{X}_2$, S would first decrease and then increase. In other words, the maximum of the absolute value of S is where the changepoint probably is located. For a graphical example see figure 14.

All CUSUMs have a maximum value but that doesn't necessary equal a changepoint. If we randomly shuffle X we get a situation where the CUSUM increases and decreases randomly and thus creating multiple local maximums or minimums. None of these are a relevant changepoint though since the data is random. The easiest way to see this is that the difference between the minimum and the maximum value is small compared to the difference for the original X. This gives us an idea how to use bootstrapping in order to decide if a sequence has a changepoint or not.

1. Calculate $S_{diff} = max(S) - min(S)$
2. Generate a bootstrap sequence, $X^{bootstrap}$, by randomly shuffle X
3. Calculate the CUSUM, $S^{bootstrap}$, of $X^{bootstrap}$
4. Calculate $S^{bootstrap}_{diff} = max(S^{bootstrap}) - min(S^{bootstrap})$
5. Redo step 2-4 a 1000 times and count how many times $S_{diff} > S^{bootstrap}_{diff}$

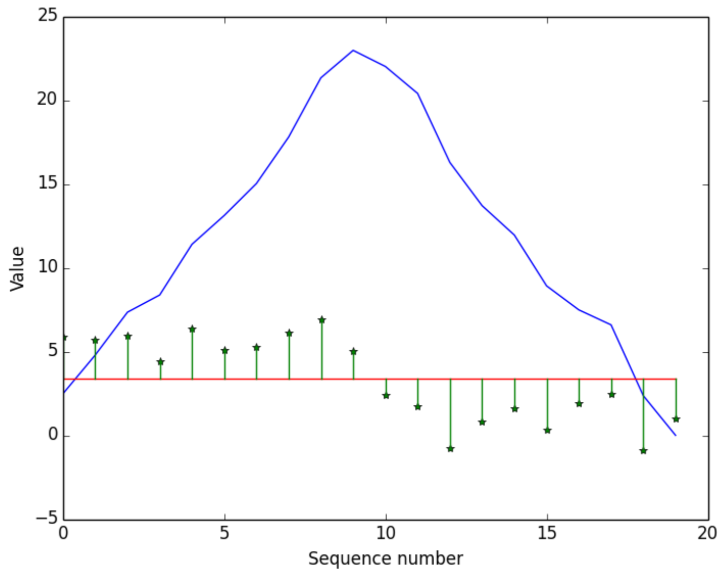

Figure 14: Example of a CUSUM for a one dimensional case. All the number, the stars, are normal distributed but the first ten number comes from a distribution with a different mean than the last ten numbers. The mean is shown as the horizontal line. For each step the difference between the mean and the current number, the vertical lines, is added to the CUSUM, the line. The top aligns with the tenth number as expected.

6. Calculate a confidence level, r, of how many times $S_{diff} > S_{diff}^{bootstrap}$

If r is larger than a certain threshold the sequence is considered to have a changepoint.

After it has been determined that a changepoint exists it also has to be found. To find a changepoint we try two different changepoint estimators. The first one has already been presented as:

$$m = \underset{i=1,2,\ldots,N}{argmax} |S_i| \tag{70}$$

That is, the point where the cumulative sum is furthest away from the mean. At this point the difference between the value and the mean must change sign in order to reach zero in the final step. A sign change at the point furthest away from the mean is probably equal to a changepoint.

A changepoint divide the sequence into two smaller sequences. One method to choose were to place the changepoint is to look at the variance in the two sequences. By minimizing the variance in each sequence we ensure that the two sequences have optimal means. If we shift the changepoint by one from the true value, we increase the variance in both sequences as well as decrease or increase the mean value in one of the sequences. Since we want to decrease the overall variance we simply sum the variance of the two sequences and find the point that minimizes this sum,

$$m = \underset{i=1,2,...N}{argmin}\Big(\sum_{i=1}^{m}(X_i - \bar{X}_1) + \sum_{i=m+1}^{N}(X_i - \bar{X}_2)\Big) \tag{71}$$

where

$$\bar{X}_1 = \frac{\sum_{i=1}^{m} X_i}{m} \quad and \quad \bar{X}_2 = \frac{\sum_{i=m+1}^{N} X_i}{N-m}. \tag{72}$$

So far $\boldsymbol{X}$ has been a vector of length N but we are interested in the case when $\boldsymbol{X}$ is a matrix of size $N \times K$. That is each X_i is actually a vector of size K. We are interested in changes that take all these dimensions into account. We do this by calculating the CUSUM separately, as explain above, for each dimension, we then sum the absolute value of each CUSUM into one final CUSUM. We are not interested if the distance to the mean is negative or not but rather the size of this distance. Summing all the CUSUM into one gives us the total distance to the means, thus giving us a way to detect changes in this distance. Finally we detect a changepoint in the same way as described above.

Once we have detected a changepoint the sequence is split into two smaller sequences. The same CUSUM-method is carried out on both sequence and if necessary they are divided into even smaller sequences. Each changepoint is saved and sorted depending on the certainty on each changepoint. When no more changepoints can be found the algorithm stops and return a vector with all the valid changepoints.

3.8 Evaluation methods

3.8.1 Confusion matrix

A confusion matrix is a way to visualize the performance of a classification method. It takes the class prediction for each observation and compare it with the observation's true class. This is done for all the observation and the result is collected in a confusion matrix.

In the two-class case the confusion matrix is made up by a 2×2 matrix in the following way:

Actual \ Predicted	D=Positive	D=Negative
C=Positive	True Positive (TP)	False Negative (FN)
C=Negative	False Positive (FP)	True Negative (TN)

Table 1: Confusion matrix.

The observations has two possible classes, "positive" and "negative". This give us four different outcomes:

True Positive (TP) = The observation predicted the "positive" class when the actual class also was "positive".

False Positive (FP) = The observation predicted the "positive" class when the actual class was "negative".

False Negative (FN) = The observation predicted the "negative" class when the actual class was "positive".

True Negative (TN) = The observation predicted the "negative" class when the actual class also was "negative".

The confusion matrix helps us define the following probabilities:

$$P(D = Positive|C = Positive) = \frac{TP}{TP + FN} \tag{73}$$

$$P(D = Negative|C = Positive) = \frac{FN}{TP + FN} \tag{74}$$

$$P(D = Positive|C = Negative) = \frac{FP}{TN + FP} \tag{75}$$

$$P(D = Negative|C = Negative) = \frac{TN}{TN + FP} \tag{76}$$

Using the confusion matrix we can create a couple of key indicators:

$$Recall = \frac{TP}{TP + FN} \tag{77}$$

Recall is the ratio of how many of the true positive the classification method found. It gives us a measurement of how many true values we miss and the goal is of course to not miss any value.

$$Precision = \frac{TP}{TP + FP} \tag{78}$$

Precision gives us a ratio of how many of our positive predicted values are actual correctly predicted. It gives us a measurement of how sure we can be on our predicted values.

Precision and recall are connected, we could easily achieve a perfect recall ratio by simply predict all values to be positive. This would hurt the precision ratio since the false positive values would increase.

$$F1 - score = \frac{2TP}{2TP + FP + FN} \tag{79}$$

The *F1-score* is the harmonic mean of the precision and recall. It weights the two errors equally and a large value mean low errors in both the recall and the precision.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{80}$$

Accuracy is a measurement using the complete matrix giving us a ratio of how correct the overall performance of the classification algorithm. That is, what is the probability for us being correct.

Indoor/outdoor classification tend to have a rather even distribution between the two classes and the distinction between the two are relatively clear. An image can be classified as either indoors or outdoors thus making it relevant to talk about recall and precision.

For the segmentation problem it is not quite as easy to define two different classes. We have a lot more locations where no segmentation should be performed than we have segmentation points. It depends on the situation but on average more than a hundred image are collected into a segment. This causes some issues with the normal definition of the confusion matrix. Even a small error when selecting a segmentation point might cause a false positive/false negative pair. In order to avoid this we accept small errors between the real segmentation point and the predicted one. We construct an *alternative confusion matrix* with the following definitions for the segmentation problem:

True Positive (TP) = An estimated segmentation location is within 5 images of a real segmentation location that has not already been used.

False Positive (FP) = An estimated segmentation location is not within 5 images of a real segmentation location that has not already been used.

False Negative (FN) = A real segmentation location is not within 5 images of an estimated segmentation location that has not already been used.

The "has not already been used" part is used to ensure that a location is not counted multiple times. This could otherwise happen if we would have two segmentation locations within 5 images of each other. For example consider a situation where two estimated segmentation points are close to each other and close to a real segmentation point. In a case like this we would not want both of the estimated points to be considered true positives. Instead we would want to consider one of the points as true positive while the other should be considered a false positive.

This definition has a major issue, we are unable to define true negative locations, we do not know which estimated negative class correspond to the real negative class. This is the cost for accepting non-perfect segmentations but it is an acceptable cost.

3.8.2 Jaccard similarity

Jaccard index is a way to determine the similarity between two sets, A and B. It is defined as:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{81}$$

The maximum similarity is one since two exact sets have the same length, the minimum similarity is zero since the length of the intersection of two independent sets is zero.

Our segmentation sets will be made up by the segmentation point indexes, for example $[100, 200, 300]$. The sets will be small compared to how many different values the sets actually can take. Consider another set, $[99, 199, 299]$, in other words an almost identical set. This is however not reflected in the Jaccard index, non of the values in the set are the same and thus the Jaccard index is zero. The Jaccard index doesn't take any consideration of how close the values in the two set are. Thus a small error in the segmentation algorithm might alter the Jaccard index to look the same as a much larger error.

To fix this problem we alter our usage of the Jaccard index. In our context set A and B are actually sorted sets of integers. This allow us to create new sets for each consecutive pair of integers in original set. The new sets consist of the integers between and including the two values in each pair. Each new set from set A is then compared with all the new sets in set B using Jaccard index. If set B original had M values, $M-1$ comparisons will be made for each pair in set A. All of these comparisons are summed and thus we get $N-1$ sums if the length of set A is N. Finally we take the average of these sums to get a final similarity value, S.

As previously, $0 \leq S \leq 1$ since each sum will be one if the two sets are the same. This algorithm takes into account the size of an error making it suitable for our needs. The algorithm is fairly quick to compute since we don't actually need to create new sets, it can easily be computed using simple arithmetics.

3.8.3 Edit Distance

Edit distance is based on strings to evaluate how many operations that are required to make two strings equal. This method can be extended to sequences and instead measure the dissimilarity between two sequences. The idea is to punish three different operations differently, add a value, delete one or "move" one. The number and type of each operation is reflected to the dissimilarity score where a lower score is better. The problem is finding the lowest score for each sequence given the cost of each operation. In practice we also have a problem of finding well suited costs for each operation, we want a good result to be equal a low dissimilarity score and vice versa. The optimal score can efficiently be computed by a dynamic programming method called WagnerFischer algorithm. This procedure removes any duplicated calculations making it suitable even for very long sequences.

We will introduce the edit distance using an example, but the method is in no means restricted to this particular situation. Consider two integer sequences, $\boldsymbol{A} = [6, 10, 40, 53]$ and $\boldsymbol{B} = [1, 5, 35, 40, 55]$. We want to make B equal to A with the use of three operations, add a value to $\boldsymbol{B}$, remove a value or "move" a value. We construct a matrix X of size $(5) \times (6)$ that has some special properties. A movement $X(i,j) \rightarrow X(i,j+1)$ equals deleting value B_j, a movement $X(i,j) \rightarrow X(i+1,j)$ equals adding A_i and finally a movement $X(i,j) \rightarrow X(i+1,j+1)$ equals moving B_j to be equal to A_i. The first row is thus equal to deleting all the values from B and the first column to add all A's value to B. We now consider the most efficient operation to reach $X(1,1)$, that is, to make A_1 equal to B_1. For this we need to know the cost of each operation, in this thesis we chose the following:

Cost to add a value: 50

Cost to delete a value: 50

Cost to move a value from b to a: $\begin{cases} 0 & if\ |b-a| < 5 \\ |b-a| & otherwise \end{cases}$

In our example the cost of moving from B_1 to A_1 is 5 since $|6-1| = 5$. This is more efficient than deleting B_1 and then adding A_1 or to first add A_1 and then delete B_1. We thus save 5 to $X(1,1)$. Now consider the cost of moving from B_2 to A_1. This can be done in three ways,

- Delete B_1 then move B_2
- Delete B_1, delete B_2 then add A_1
- Make B_1 equal A_1 then delete B_2

Delete B_1 has already been computed in $X(0,1)$. Delete B_1, delete B_2 has already been computed as $X(0,2)$. Make B_1 equal to A_1 has already been computed in $X(1,1)$. In fact, we only need to do three calculations, $X(1,1)$ + delete B_2, $X(0,2)$ + add A_1 and $X(0,1)$ + move B_2. These procedures can be applied to the remaining parts of the matrix, we get:

$$X(i,j) = min \begin{cases} X(i,j-1) + \textit{delete cost} \\ X(i-1,j) + \textit{add cost} \\ X(i-1,j-1) + \textit{cost to move } B_i \textit{ to } A_j \end{cases} \tag{82}$$

This operation is carried out step by step for the other positions in the matrix until we reach the end, that is $X(m,n)$. This value is the optimal cost to make B equal to A. See figure 15 for the complete example.

$A\backslash B$	–	**1**	**5**	**35**	**40**	**55**
–	0	50	100	150	200	250
6	50	5	50	100	150	200
10	100	55	10	60	110	160
40	150	105	60	15	60	110
53	200	155	110	65	28	60

Figure 15: The complete example of how to calculate edit distance.

4 Indoor/Outdoor classification

The transition between indoor and outdoor plays a vital role when a sequence is segmented into moments. In most situations moving from inside to outside signifies a major change in both the environment and the activities. The same is obviously true the other way around. A large change like this is of major interest and should probably be considered to be a segmentation point. Although being a strong separation cue, care needs to be taken due to the nature of indoor/outdoor classification. The classification is insufficient in a couple of situations, it would fail to distinguish any moment that is completely indoors or outdoors. There also exist situations that should be considered to belong to the same moment even though they contain both indoor and outdoor images. In particular situations when the user is moving back and forth from an indoor and an outdoor environment.

Multiple methods have been constructed for indoor/outdoor classification [20][6][16] with various results. The majority of the methods uses different low-level features like color, textures or edges but methods using higher level features also exists. All of these algorithms are used to classify a single image rather than longer sequences. These methods could of course be used in sequence by classifying one image at a time but results could be improved by using knowledge from the previous images when classifying a new image in a sequence.

4.1 Our indoor/outdoor classification method:

Descriptors In our indoor/outdoor classification we will try multiple low level features, the RGB histogram, the HSV histogram, the LAB histogram, the ECOH features and the Texton features. We also try the SIFT features in order to test a slightly higher level feature. We compute all the features on all the images in the indoor/outdoor dataset described in section 2.2.

SVM As stated previously we select 40 % of these images and use these to train one radial basis function probabilistic-SVM for each descriptor as described in section 3.6.1.

In order to avoid overfitting we incorporated a cross-validation scheme and used a gridsearch to optimize the parameters of each SVM. The end result is one SVM per descriptor that has been trained on data separate from the evaluation data. We are interested in finding the optimal feature for indoor/outdoor classification and is thus not interested in combining multiple SVMs.

HMM Consider a sequence of images in chronological order. If the previous image was inside it is much more likely that the next image also is inside. The average day tend to have few transitions from indoor to outdoor or vice versa. Thus by looking at the previous image we usually get a reasonable guess of the state of the current image. We could of course instead use the last two images but that is not something that have been done in this thesis. The previous image provides enough information about the past without introducing too much computations making the algorithm too expensive.

The hidden markov model described in section 3.6.2 combines the likelihood estimates from the probabilistic-SVM with the probability of transitioning between states and initial state probability. The hidden states are in this context indoor and outdoor and the observations consist of our descriptors. The observation probability distributions b is the trained probabilistic-SVM method. It provides a probability of being indoors or outdoors given an observation. In order to provide a good classification the HMM requires knowledge of the state transitioning as well as the starting probability. Both of these value are estimated using the notations from all the datasets.

For the state transitions, a, we simply count the number of transitions between the states and divide by the number of transitions. That is, we count how many transitions going from indoor to indoor and indoor to outdoor and divide each by the sum of the two. The same procedure is carried out for outdoor to indoor and outdoor to outdoor. We now have a transition probability that gives us the likelihood for each transition.

The initial state probability is estimated by looking at the first image in each sequence. We count how many of these are indoors and outdoor and divide by the number of sequences. This gives us a probability of where a user starts using the clip.

The HMM provides a binary classification of the sequence. The hope is that the extra information from the previous images should improve the classification over the original stand-alone probabilistic-SVM classification.

5 Segmentation strategies

The problem of segmenting a sequence into moments is actually a two-parted problem. Given a sequence we need to find the number of moments as well as where the moment boundaries are located. In this thesis we primarily deal with the later of the two problems but they are interconnected and thus we will also touch upon the first problem. According to our moment definition a moment is coherent in the sense that the mean is fairly consistent through out the moment. A fair assumption is thus that different moments should have different means if they are to be distinguished between each other. For example, playing football should result in multiple green images while sailing should mostly consist of blue images. The green color mean of these moments are different and thus distinguishable. This argument extends to our segmentation problem, given a sequence with presumably multiple moments we should be able to find multiple segments with different means. This reasoning simplifies our problem, by finding all the place where the mean changes with a reasonable magnitude we actually find the moment boundaries that we are looking for.

The mean can be calculated in multiple ways. The goal is as stated in the moment definition that the moment is visually coherent. In this thesis we interpret that as taking the mean of the descriptors used to describe the images. A combination of these descriptors could be used to find changes in multiple different image characteristics.

Defining a good segmentation strategy will require optimizing multiple parameters, such as sensitivity to mean changes and how to consider different descriptor. For this we use he images described in section 2.2.

We arrive at the following segmentation strategy where part 1. to 5. only needs to be run once.

1. **Descriptors**
   Calculate all the descriptors for the training set.

2. **Indoor classification**
   Classify all the image in the training set as either indoor or outdoor.

3. **Changepoints**
   Calculate the changepoints at various sensitivity for each sequence and descriptor separately.

4. **Optimize segmentation points**
   Select the best settings in order to optimize the segmentation points for a specific measurement metric.

5. **Combine segmentation points**
   Combine the indoor classification and the segmentation points from

the multiple descriptors into a single set of segmentation points optimized for a specific measurement metric.

6. **Segment new sequence**
   Calculate the descriptors and indoor classification, calculate the segmentation points using the optimal settings and combine them using the optimal combining procedure.

5.1 Descriptors used

In this thesis we will try the following descriptors:

- Color variations will be detected using three different 192-bin color histograms as defined in section 3.5.1, that is a RGB-histogram, a HSV-histogram and a LUV-histogram.
- Texture changes will use the Texton descriptors as defined in section 3.5.6
- Two different global scene descriptor will be used, Centrist described in section 3.5.4 and GIST outlined in section 3.5.3
- The ECOH descriptor, section 3.5.2, will be used to detect changes in a combination of color and lines.
- Finally we hope to detect object distribution changes with the SIFT descriptor as described in section 3.5.5

5.2 Indoor/outdoor segmentation

All the images are classified as either indoors or outdoors using the HMM model described in section 4. The actual class of the images is not of interest here, instead we are interested in when the classes changes. That is the transition from indoors to outdoors or outdoors to indoors. These transition are located and saved as the indoor/outdoor segmentation points. Our assumption is that these locations has a high correlation with the actual segmentation points.

5.3 Changepoints

Estimating the changepoints will be done by the method described in section 3.7. This method takes a sequence of multivariate data, that is our descriptors, and finds the points where the mean changes. The certainty for a change in mean needs to be above a certain confidence value in order to be determined significant. We will try three different confidence values, 85%, 90% and 95%. In order to actually find the changepoints we try the two different estimators outlined in section 3.7. Finally we also want to choose the maximum number of changepoints in a sequence. Certain days might have more changes in mean than what is reasonable. In these cases we want to select the most significant changepoints and remove the rest. Longer sequences can have more changepoints than smaller ones simply because more time allows for more changes. The maximum number of changepoints is determined using a ratio of the total number of images in a sequence. In our case we limit the number of changepoints to either 0.5%, 1.0%, 1.5% or 2.0% of the sequence length. All the remaining changepoints are then considered to be estimated segmentation points.

To summarize, the segmentation points are dependent on the parameter $\theta = (confidence\ level,\ changepoint\ estimator,\ maximum\ ratio)$.

5.4 Segmentation point optimization

In order to optimize the parameter θ we need a measurement to determine the performance. Two such metrics have been presented, the Jaccard similarity in section 3.8.2 and the edit distance in section 3.8.3. Both of these distance measurements generate a similarity measurement for a sequence given a set of estimated segmentation points and the groundtruth segmentation points.

We calculate the distance measurements for each sequence. We then calculate the average of these numbers and get two different total similarity value, one for each metric, that determines the overall performance of the parameter θ. For the Jaccard similarity a higher number is better while the opposite is true for the edit distance.

The actual optimization is done by comparing the total similarity measurement for all the possible values of θ. The θ with best total similarity measurement is selected. This is done for all the descriptors resulting in 16 different values of θ (eight descriptors with a different θ for each of the two similarity measurement).

5.5 Combine descriptors

A human uses a wide range of cues to determine what an image describes and allows us to distinguish a wide range of situations and objects. In order to some extent mimic this behavior we want to combine our descriptors in order to capture different aspects of the images. We will try to combine the descriptor using three different methods:

5.5.1 Add descriptors

The easiest method is to simply add the descriptors' segmentations and use the sequence averaging method described in section 3.1. We optimize the threshold using all the photo-sets and the distance measurements. In other words we require a certain amount of different segmentation to agree on a specific point in order to select it as a combined segmentation point.

5.5.2 Probability of a segmentation point

Some of the descriptors might perform better than others and we it would be beneficial to incorporate this behavior when combining the descriptors. This can be done by taking a probabilistic approach to the combining problem:

For a certain location we want to determine the probability for a segmentation point given how the descriptors estimates the specific point, mathematically it can be written as

$$P(S|D_1, D_2, ..., D_N). \tag{83}$$

S stands for segmentation point and indicates the interest of a segmentation point. D_i indicates the prediction of the descriptors, that is if $D_i = 1$ it would indicate that the i:th descriptor believe the location to have a segmentation point. We can rewrite this as

$$P(S|D_1, D_2, ..., D_N) = \frac{P(S)P(D_1, D_2, ..., D_N|S)}{P(D_1, D_2, ..., D_N)}. \tag{84}$$

Changing S doesn't change the indications from the descriptor and we can reduce further to

$$P(S|D_1, D_2, ..., D_N) \propto P(S)P(D_1, D_2, ..., D_N|S). \tag{85}$$

Using naive conditional independence we can simplify our algorithm. This is a simplification in order to remove any joint probability distribution. This could be a good simplification if the correlation between the descriptors is small, that might not be the case here since multiple descriptors relies on colors. But due to time-constraints we are unable to construct a proper joint probability distribution.

$$P(S|D_1, D_2, ..., D_N) \propto P(S)P(D_1|S)P(D_2|S) \cdot ... \cdot P(D_N|S) \tag{86}$$

Finally we take logarithm and arrive at the final equation

$$P(S|D_1, D_2, ..., D_N) \propto ln(P(S)) + \sum_{i=1}^{N} ln(P(D_i|S)). \tag{87}$$

To fully describe this equation we need four values for each descriptor, $P(D_i = 1|S = 1)$, $P(D_i = 0|S = 1)$, $P(D_i = 1|S = 0)$ and $P(D_i = 0|S = 0)$ where one represents the presences of a segmentation point while zero does not. As described in section 3.8.1 our alternate definition of the confusion matrix makes it troublesome to define the last probabilities satisfactory.

That is, we are able to define the probability that a point is a segmentation point but not able to define that a position is not a segmentation point.

$$\begin{aligned} P(D_i = 1|C = 1) &= \frac{true\ positive}{true\ positive + false\ negative} \\ P(D_i = 0|C = 1) &= \frac{false\ negative}{true\ positive + false\ negative} \end{aligned} \tag{88}$$

The result is not complete useless, we can use this result as an indicator for a position to be a segmentation point and simply threshold it in order to actually make the decision.

5.5.3 F1-score

The probability combination method does not take any consideration to the false positive segmentation points as can be seen in equation . This mean that this method might prioritize descriptors that select multiple segmentation points rather than correct ones. The F1-score as described in section 3.8.1 takes both the false positives as well as the false negative into account and thus give us a value that weight the two errors equally. Instead of simply taking sum of all the descriptors as we do in section 5.5.1 we take the sum of each descriptor's F1-score. That is each position where a descriptor estimates a segmentation point is now weighted with the F1-score and all other positions have value of zero. We then threshold this value like we did in the add combination method by using the sequence averaging method. This method doesn't have as strong theoretical background as the probability method but does on the other hand not favor descriptors with an abundance of false positive segmentation points.

6 Evaluation

6.1 Human segmentation points

The human segmentation points are mostly used to create a groundtruth for the computer based algorithms. But it can also be used to understand the complexity of the problem and give us a baseline to compare against. That is why we have selected the best and worst humans compared to the groundtruth and presented the results for the ”Summerday” photoset.

Alternative confusion matrix for ”Summerday”

Edit		
	Best	Worst
True positive	9	10
False positive	3	20
False negative	1	0

Jaccard		
	Best	Worst
True positive	9	2
False positive	3	1
False negative	1	8

Table 2: Alternative confusion matrix for the best and worst human segmentation compared to the groundtruth using the ”Summerday” photoset. The two distance measurements prioritize differently, we show both.

Keyvalues for ”Summerday”

Edit		
	Best	Worst
Edit distance	105	1000
Recall	0.9	1
Precision	0.75	0.3
F1-score	0.82	0.5

Jaccard		
	Best	Worst
Jaccard similarity	0.92	0.27
Recall	0.9	0.2
Precision	0.75	0.67
F1-score	0.82	0.31

Table 3: The different keyvalues and measurement distances for the best and worst human segmentation compared to the groundtruth using the ”Summerday” photoset.

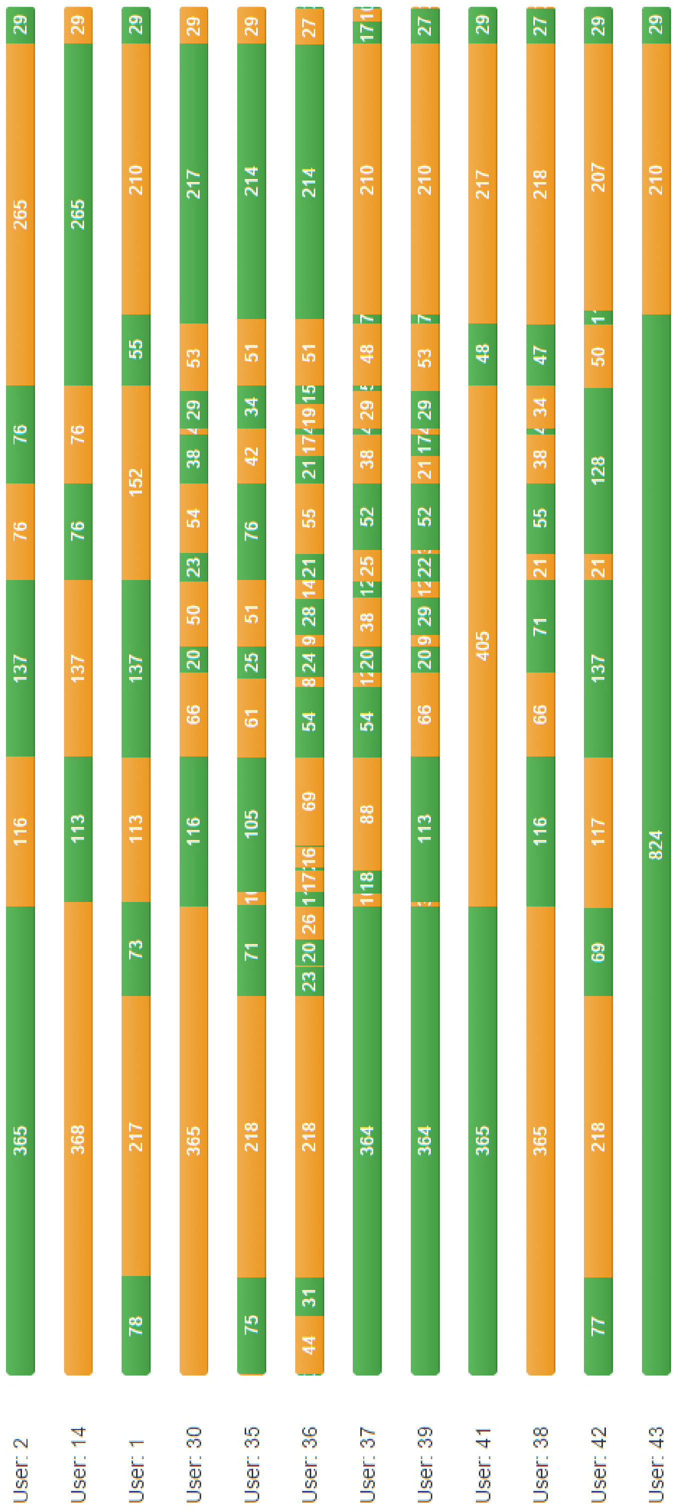

Figure 16: A graphical view of how the different users segmented the "Summerday" photoset. Each block represent a, according to the user, coherent segment. The number inside each block is the length of the block. The first four users, the is user 2,14,1 and 30 are all experienced with the Narrative Clip and its usage. The remaining users have limited or no experience with the Clip.

6.2 Indoor/outdoor classification

The indoor classification is the foundation to the indoor/outdoor segmentation algorithm. Here we present the results for both the SVM and the HMM giving us a way to compare the effect of adding the HMM.

Mean accuracy

	SVM	HMM
RGB	88%	94%
HSV	93%	97%
LAB	83%	94%
ECOH	84%	95%
Texton	89%	97%
SIFT	88%	96%

Table 4: The accuracy when classifying the images as either indoor or outdoors using different descriptors and classification methods.

To get a better understanding of both the difference between different descriptors and the effect of the HMM we present the confusion matrices for the two best descriptors, the HSV and the Texton, as well as one of the worst descriptors, the RGB.

Confusion matrix HSV

SVM		
True \ Predicted	Indoor	Outdoor
Indoor	11753	744
outdoor	840	10702

HMM		
True \ Predicted	Indoor	Outdoor
Indoor	12174	323
Outdoor	354	11188

Table 5: The confusion matrix for the indoor/outdoor classification using the HSV descriptor, here presented using the two different classification methods.

Confusion matrix Texton

SVM		
True \ Predicted	Indoor	Outdoor
Indoor	11208	1289
outdoor	1380	10162

HMM		
True \ Predicted	Indoor	Outdoor
Indoor	12014	483
Outdoor	349	11193

Table 6: The confusion matrix for the indoor/outdoor classification using the Texton descriptor, here presented using the two different classification methods.

Confusion matrix SIFT

SVM		
True \ Predicted	Indoor	Outdoor
Indoor	11276	1221
outdoor	1588	9954

HMM		
True \ Predicted	Indoor	Outdoor
Indoor	12101	396
Outdoor	469	11073

Table 7: The confusion matrix for the indoor/outdoor classification using the SIFT descriptor, here presented using the two different classification methods.

To visualize the issues with the SVM method as well as the HMM method we present the same descriptors above except now as a visualization of the indoor/outdoor segmentation of the "Summerday" photoset.

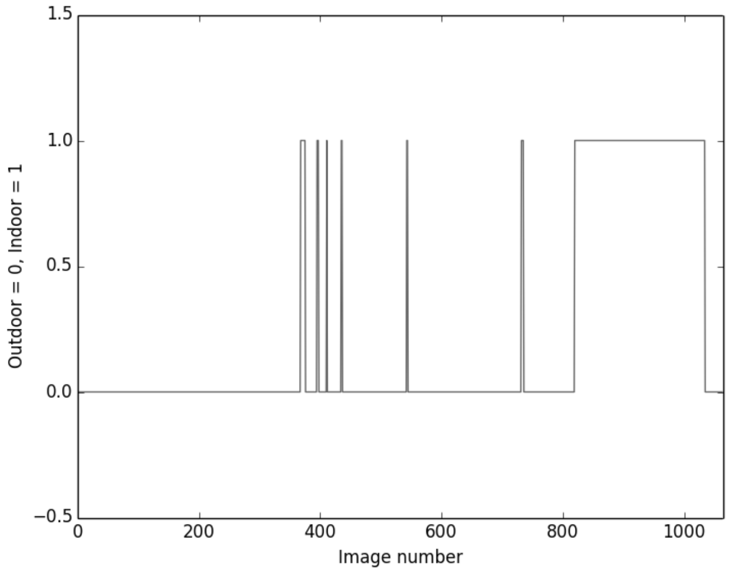

Figure 17: The true indoor/outdoor classification of the "Summerday" photoset as annotated by a human.

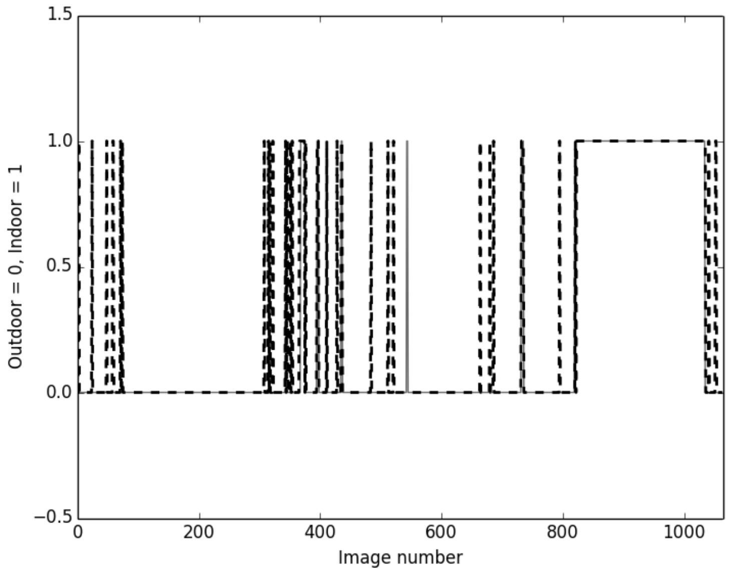

Figure 18: The indoor/outdoor classification for the "Summerday" photoset using the HSV descriptors and the SVM classification method. The dotted line is the HSV-SVM method while the filled line is the true classification.

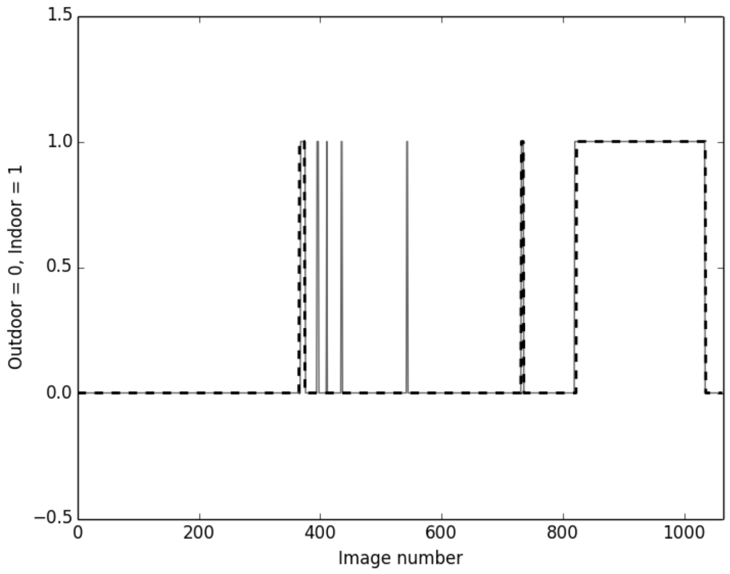

Figure 19: The indoor/outdoor classification for the "Summerday" photoset using the HSV descriptors and the HMM classification method. The dotted line is the HSV-HMM method while the filled line is the true classification.

6.3 Indoor outdoor as segmentation points

In order to see how the indoor/outdoor classification work as segmentation points we create the alternative confusion table, presented in section 3.8.1, using all the photo-sets.

Alternate confusion matrix

True positive	100
False positive	179
False negative	76

Table 8: The alternate confusion matrix calculated for the HSV-HMM indoor/outdoor segmentation. All photo-sets were used, that is, the alternate confusion matrix for each photoset was summed.

Keyvalues and distance measurements

Mean Edit distance	324
Mean Jaccard similarity	0.64
Recall	0.57
Precision	0.36
F1-score	0.44

Table 9: The keyvalues and distance measurements for the HSV-HMM indoor/outdoor segmentation using all the photo-sets.

To visualize the pros and cons of this method, we present the groundtruth segmentation points as well as the indoor/outdoor segmentation points with the norm of the RGB descriptor as a basic reference.

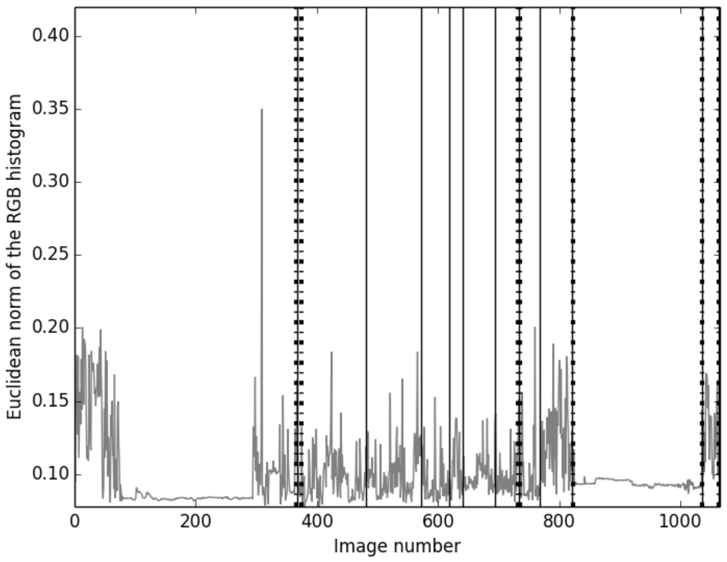

Figure 20: The HSV-HMM indoor/outdoor segmentation are visualized as the dotted vertical lines, while the true segmentation are the filled vertical lines. The graph represent the euclidean norm of the RGB descriptors, this is used in order to get some understanding of how system develops.

6.4 Segmentation points optimzed over all photo-sets

The changepoint analysis is calculated for each descriptor and photoset. The mean value of the two distance measurements are calculated and presented:

Average distance measurements

	Edit distance	Jaccard similarity
RGB	241	0.54
HSV	219	0.57
LAB	221	0.54
ECOH	216	0.57
Texton	221	0.56
GIST	227	0.59
SIFT	216	0.57
Centrist	221	0.55

Table 10: The average distance measurements for the different descriptors changepoint segmentations calculated using all photo-sets.

To better understand the consequence of the different distance measurements the alternate confusion matrix is populated. Using this matrix we also calculate the key values to get an understanding of the performance for each descriptor and distance measurement.

Alternate confusion matrices

Edit changepoints			
	TP	FP	FN
RGB	31	74	145
HSV	44	61	132
LAB	30	75	146
ECOH	34	71	142
Texton	31	74	145
GIST	30	74	146
SIFT	33	72	143
Centrist	35	70	141

Jaccard changepoints			
	TP	FP	FN
RGB	53	151	123
HSV	64	146	112
LAB	48	167	128
ECOH	53	165	123
Texton	51	175	125
GIST	47	158	129
SIFT	53	167	123
Centrist	59	168	117

Table 11: The alternate confusion matrices for the different descriptors changepoint segmentations calculated using all photo-sets.

Keyvalues

Edit changepoints			
	Recall	Precision	F1-score
RGB	0.18	0.30	0.22
HSV	0.25	0.42	0.31
LAB	0.17	0.29	0.21
ECOH	0.19	0.32	0.24
Texton	0.18	0.30	0.22
GIST	0.17	0.29	0.21
SIFT	0.19	0.31	0.23
Centrist	0.20	0.33	0.25

Jaccard changepoints			
	Recall	Precision	F1-score
RGB	0.30	0.26	0.28
HSV	0.36	0.30	0.33
LAB	0.27	0.22	0.25
ECOH	0.30	0.24	0.27
Texton	0.29	0.23	0.25
GIST	0.27	0.23	0.25
SIFT	0.30	0.24	0.27
Centrist	0.34	0.26	0.29

Table 12: The keyvalues for the different descriptors changepoint segmentations using all the photo-sets. Caluclated from table 11.

Instead of calculating the above values on all photo-sets we can also calculate the same information using only the "Summerday" photoset.

Distance measurements for "Summerday"

	Edit distance	Jaccard similarity
RGB	572	0.38
HSV	619	0.32
LAB	563	0.37
ECOH	359	0.42
Texton	364	0.32
GIST	250	0.32
SIFT	452	0.34
Centrist	486	0.39

Table 13: The distance measurements for the different descriptors changepoint segmentations calculated using the "Summerday" photoset.

Alternate confusion matrix for ”Summerday”

Edit changepoints			
	TP	FP	FN
RGB	0	5	10
HSV	1	4	9
LAB	1	4	9
ECOH	3	2	7
Texton	3	2	7
GIST	4	1	6
SIFT	2	3	8
Centrist	1	4	9

Jaccard changepoints			
	TP	FP	FN
RGB	0	8	10
HSV	1	7	9
LAB	2	8	8
ECOH	3	2	7
Texton	3	7	7
GIST	3	5	7
SIFT	2	8	8
Centrist	3	7	7

Table 14: The alternate confusion matrices for the different descriptors changepoint segmentations calculated using the ”Summerday” photoset.

Keyvalues for ”Summerday”

Edit changepoints			
	Recall	Precision	F1-score
RGB	0	0	0
HSV	0.10	0.20	0.13
LAB	0.10	0.20	0.13
ECOH	0.30	0.60	0.40
Texton	0.30	0.60	0.40
GIST	0.40	0.80	0.53
SIFT	0.20	0.4	0.27
Centrist	0.10	0.20	0.13

Jaccard changepoints			
	Recall	Precision	F1-score
RGB	0	0	0
HSV	0.10	0.13	0.11
LAB	0.20	0.20	0.20
ECOH	0.30	0.60	0.40
Texton	0.30	0.50	0.38
GIST	0.30	0.38	0.33
SIFT	0.20	0.20	0.20
Centrist	0.30	0.30	0.30

Table 15: The keyvalues for the different descriptors changepoint segmentations using the ”Summerday” photoset. Caluclated from table 14.

To better visualize the difference between the different descriptors we also present the actual segmentation of the "Summerday" photoset. We present the results as both indexes as well as in graphs using the corresponding norm as a baseline of how the image sequence changes.

Segmentation points for "Summerday"

	Edit	Jaccard
RGB	45 74 294 353 776	27 45 74 294 353 591 635 776
HSV	46 76 294 298 824	39 46 76 294 298 375 649 824
LAB	42 76 293 354 824	27 42 61 76 200 293 354 477 598 824
ECOH	79 369 472 769 824	79 369 472 769 824
Texton	76 363 495 770 824	46 76 184 294 363 399 495 770 802 824
GIST	79 363 617 769 824	27 46 78 294 343 368 690 823
SIFT	78 296 641 822 1004	44 78 165 211 296 476 641 822 940 1004
Centrist	78 363 544 788 947	44 78 245 294 363 479 544 788 824 947
Groundtruth	367 481 572 618 641 694 734 768 821 1035	

Table 16: The actual segmentation points as estimated by each descriptor as well as the true segmentation points for the "Summerday" photoset.

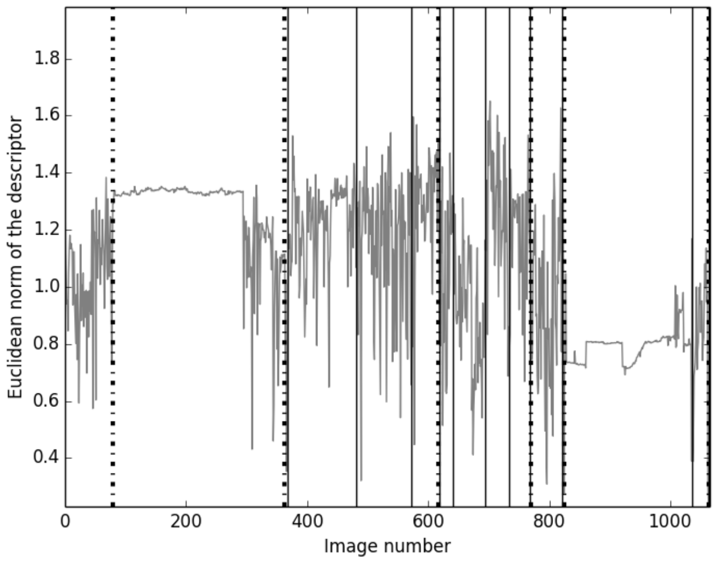

Figure 21: Visulization of the GIST segmentation points calculated using the edit distance measurement and the "Summerday" photoset. This segmentation is the dotted vertical lines while the true segmentation is the filled vertical lines. The graph is the euclidean norm of the GIST descriptor.

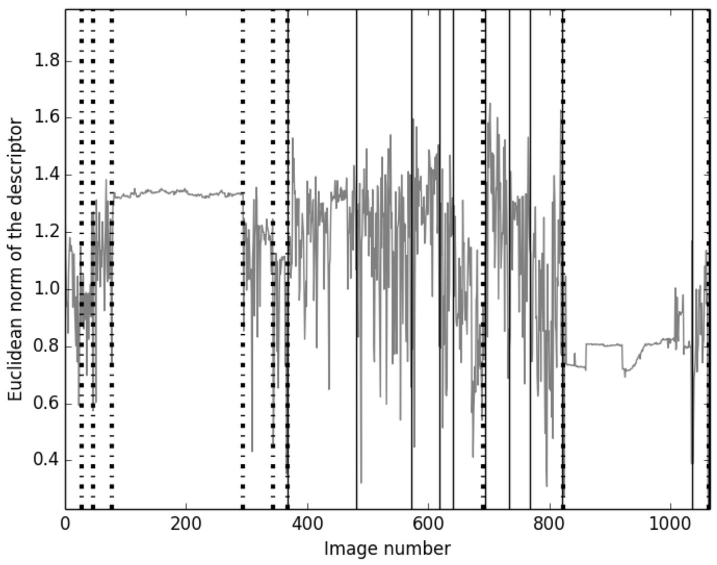

Figure 22: Visulization of the GIST segmentation points calculated using the Jaccard similarity measurement and the "Summerday" photoset. This segmentation is the dotted vertical lines while the true segmentation is the filled vertical lines. The graph is the euclidean norm of the GIST descriptor.

6.5 Combined segmentation points optimzed over all photo-sets

The combined segmentation points are presented much in the same way as the previous section with the difference that the descriptors have been replaced with the different combination methods.

Distance measurements

	Edit distance	Jaccard similarity
Add	208	0.62
Prob	229	0.58
F1-prob	185	0.64

Table 17: The average distance measurements for the different combination methods' segmentations calculated using all photo-sets.

Alternate confusion matrix

Edit changepoints			
	TP	FP	FN
Add	62	87	114
Prob	34	85	142
F1-prob	52	49	124

Jaccard changepoints			
	TP	FP	FN
Add	93	199	83
Prob	65	221	111
F1-prob	79	131	97

Table 18: The alternate confusion matrices for the different combination methods' segmentations calculated using all photo-sets.

Keyvalues

Edit changepoints			
	Recall	Precision	F1-score
Add	0.35	0.42	0.38
Prob	0.19	0.29	0.23
F1-prob	0.30	0.51	0.38

Jaccard changepoints			
	Recall	Precision	F1-score
Add	0.53	0.32	0.40
Prob	0.37	0.23	0.28
F1-prob	0.45	0.38	0.41

Table 19: The keyvalues for the different combination methods' segmentations using all the photo-sets. Caluclated from table 18.

As before we present the results for the "Summerday" photoset since it gives us a way to visualize and analyze the behavior of the different methods.

In the following figures the dotted vertical lines represent the method's segmentation points while the filled vertical lines represent the groundtruth's segmentation points. To get some understanding of where in the system a certain segmentation point take place we show the euclidean norm of the RGB descriptor.

Keyvalues for "Summerday"

	Edit distance	Jaccard similarity
Add	555	0.40
Prob	514	0.53
F1-prob	400	0.32

Table 20: The distance measurements for the different combination methods' segmentations calculated using the "Summerday" photoset.

Alternate confusion matrix for "Summerday"

Edit changepoints			
	TP	FP	FN
Add	2	5	8
Prob	1	5	9
F1-prob	3	1	7

Jaccard changepoints			
	TP	FP	FN
Add	2	7	8
Prob	2	9	8
F1-prob	2	4	8

Table 21: The alternate confusion matrices for the different combination methods' segmentations calculated using the "Summerday" photoset.

Keyvalue for "Summerday"

Edit changepoints			
	Recall	Precision	F1-score
RGB	0.20	0.29	0.24
Prob	0.10	0.17	0.13
F1-prob	0.30	0.75	0.43

Jaccard changepoints			
	Recall	Precision	F1-score
RGB	0.20	0.22	0.21
Prob	0.20	0.18	0.19
F1-prob	0.20	0.33	0.25

Table 22: The keyvalues for the different combination methods' segmentations using the "Summerday" photoset. Caluclated from table 21.

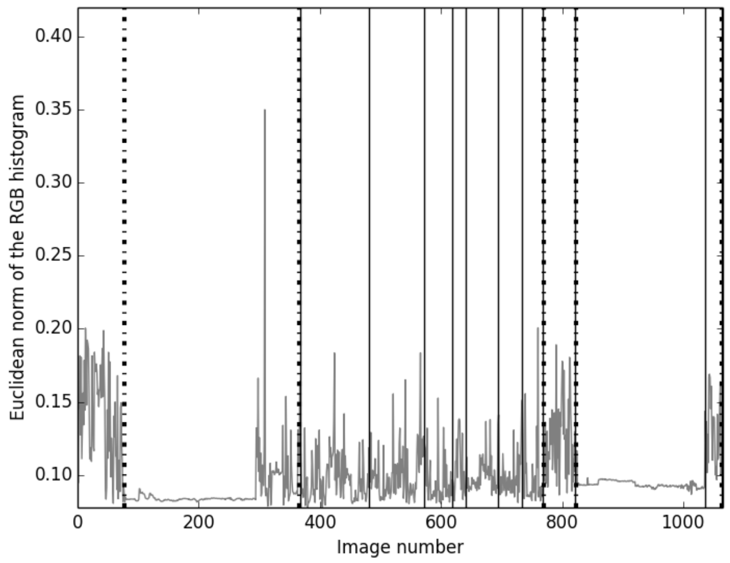

Figure 23: Visulization of the F1-score combination method calculated using the edit distance measurement and the "Summerday" photoset.

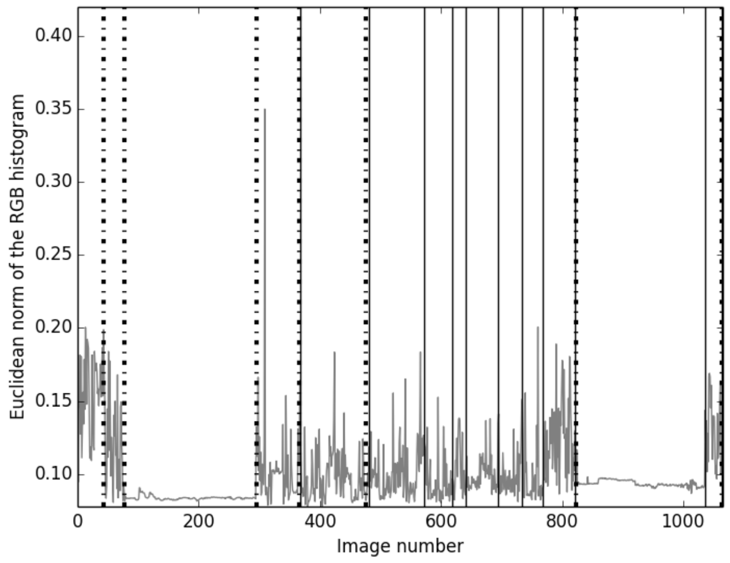

Figure 24: Visulization of the F1-score combination method calculated using the jaccard distance measurement and the "Summerday" photoset.

7 Discussion

7.1 Human segmentation points

At a first glance it sounds pretty easy to divide a day into moments. First you do one thing and then you do another and the distinction between the two is natural. To some extent that is actually true, dividing your own images is not a complicated process. You have experienced the day, you know which parts are connected even though the images might not show it. There exists some situations that might be somewhat unclear but those are few in comparison. The situation becomes a lot more complex when you ask other people to do the segmentation. In figure 16 we see a wide range of segmentations, some people segment rather coarsely while others are a lot more precise. In part this a result of somewhat unclear instructions as well as unfamiliarity with the Narrative Clip. Perhaps a better measurement would be the users that were familiar with the Clip, these users are also shown in figure 16. The users have less variations but there is still quite a lot of variations.

All though the variations are large we can still find similarities. At some locations almost all users agree with each other and at other locations most of the users agree. These are the locations that have been detected when we created the groundtruth using the sequence averaging method. Comparing the humans' segmentation we notice that the result is not very good, even the best human only has a precision of 0.75. That is every fourth segmentation point is not actually a real segmentation point.

These results should of course be used with great care due to the lack of testing data but nevertheless it gives an indication to hard this problem actually is, even for humans.

7.2 Indoor/Outdoor classification

The indoor/outdoor classification using only the SVM method performs very well, it is better than most equivalent algorithms presented in previous papers. In fact, it is probably too good. In this thesis we have used data collect by two people doing a limited range of activities. Due to the image taking multiple pictures at the same location further limits the variety in the training data. The reality is much more complex and the algorithm is probably severely over-fitted even though care was taken to prevent this.

The different descriptors varies quite substantially, up to as much as 10%. This is to be expected as some algorithm are more complex than others. What was not expected is that the LAB-histogram performed worse than both the RGB and the HSV. The LAB was constructed to represent human vision better than the other two yet it performs worst of all the descriptors. Another interesting observation is that the ECOH performs worse than the HSV even though the HSV, in part, is used in the ECOH.

The most reasonable explanation is that the HSV must carry important information in the value channel that is not used in the ECOH.

Adding the HMM improves the result and reaching as high has 97% for some descriptors. 97% would be a very good result for an indoor/outdoor classification but in this case the algorithm is not used on separate images and can not be compared with standalone algorithms. We have not found any previous work on this kind of problem and thus can not compare the method with alternative approaches. That said, 97% can be understood as one wrong classified image out of 33 images. This is a very good result and even though these algorithms are over-fitted the method should provide satisfactory results.

In fact, for our application the actual result may be even better than 97%. Consider the following example, you are playing football in the garden and go inside for a quick drink. During this time the camera takes a couple of photos indoors before you go back out and continue playing. This moment can be considered irrelevant when segmenting a long day but would be captured by a perfect indoor/outdoor segmentation. An HMM trained like ours might actually remove these results due to the small likelihood to change from indoors to outdoors. In other words, the error of a HMM is usually clustered and might actually in some situations improve our segmentation results.

7.3 Indoor/outdoor classification as segmentation points

Using the indoor/outdoor classification as segmentation points is not optimal. It is very easy to generate false positives as can be seen table 8. The issue is that some situations might be counted twice. Consider for example figure 19 in which a short segment of images around index 370 are correctly classified as indoors. But when we study figure 20 we see that it actually generates two segmentation points, one going from outdoors to indoors and the other from indoors to outdoors. One of these are correct but the other one is a false positive generated by the indoor/outdoor segmentation. These errors are quite frequent in this segmentation algorithm as can be seen in table 8, the false positives are a lot more frequent than the false negatives. The false negatives arise because of segmentation points that are located completely indoors or outdoors, the indoor/outdoor segmentation algorithm can not simple capture these points. All that said, there is definitely a correlation between indoor/outdoor changes and the actual segmentation points.

7.4 Segmentation points

Looking at the performance measurements in table 10 we notice that the difference between the descriptors are rather small, at least when we compare the average distance measurements. The Texton and SIFT descriptors are slightly better when using the edit distance while the GIST descriptor is the better one using the Jaccard similarity. The difference is well inside the margin of error and we can't really make any quality assumptions among the different descriptors except possibly that the RGB descriptor is not as good as the others.

The alternate confusion matrix in table 11 and its keyvalues in table 12 gives us bit more information. The HSV has more true positives, fewer false positive and fewer false negative than any other descriptor no matter which distance measurement that was used. This of course reflects in the performance of the recall, precision and F1-score. This is actually somewhat strange, the HSV's distance measurements are worse than the SIFT yet the confusion matrix is better. This must mean that the HSV has a tendency to find segmentation points that are far away from the groundtruth segmentation points.

Another interesting behavior is the difference between the Edit distance and the Jaccard similarity. The F1-score is very similar between the two but the recall of one is similar in size to the precision of the other. This can be understood better in table 11, the Jaccard is much more forgiving to false positives and is much less conservative with its segmentation. This has the benefit of finding more true positives and by that decreasing the false negatives but the consequences are that we also find more false positives.

Applying these algorithms to the "Summerday" photoset we see a steep increase in the distance measurements. This photoset is fairly large and larger photo-sets are more prone to errors due to the length increasing the likelihood for false segmentation points. As such the increased error is expected. What is more surprising is which descriptors that changed the most, suddenly the HSV is the worst and the GIST is the best descriptor. Due to the lack of data it is hard to tell if this a normal result or just an outlier. A potential explanation could be that the HSV's tendency to select false segmentation points is more prominent in longer photo-sets than in shorter ones. It could be a result of the optimization that affect the HSV more than the GIST.

Comparing the groundtruth with the estimated results in table 16 we see a couple of things. At certain locations almost all of the descriptors agree with each other, for example at 824 all but two descriptors agree with each other. What is more interesting is that these points are very close to some of the groundtruth's values. There is a strong correlation between the groundtruth and when almost all of the descriptors agree with each other. There exists some points where this is not true, for example index 79 can be found in most descriptors yet it can not be found in the groundtruth. But if we take a closer look at figure 16 we see that four people actually thought that point 79 was a real segmentation point.

A manual comparison between the edit distance and the jaccard similarity tend to favor the former. But the results are subjective and only based on the "Summerday" photoset, thus no major distinction can really be made.

7.5 Combined segmentation points

Combining the descriptors improves the distance measurement results with up to $14 - 20\%$ which is a substantial amount. The improvement is even greater in the alternative confusion matrix. The true positives are improved by 18% and at the same time the false positives have decreased by approximately 20%. Altogether increasing the F1-score by as much as 22%. Combining the segmentation points is an important task that improves the results making it worth the extra computation time. It is however important to use the correct combination method. As can be seen in table 17 the quasi-probabilistic method didn't improve the results at all while both the F1 and the add combination methods improved the results. This is to be expected, the quasi-probabilistic method lacked any information about the false positives due to the limitations in the alternate confusion matrix.

Although the F1 combination method improves the result for the "Summerday" photoset it is hard to actually see any improvement. This is expected since the combination method is built on the descriptor segmentation points. If descriptors struggle to find most segmentation points, the combination m that have not been discovered by the descriptor segmentations. The combination methods merely improve the results by limiting the amount of false positives.

That said, we should be able to improve the result quite a bit. The F1 combination method is the best method we tried but it does not have a solid theoretical background. The optimization was done in parts. First we optimized each descriptor separately and then we optimized the combination method. This could be combined into one single optimization method and which would probably improve the result further but this is a time-consuming task and has not been done in this thesis.

8 Conclusion

The overall results are on average quite bad. Our best segmentation is done by the F1 combination method and we still only reach a recall of 0.3 with a precision of 0.51. This means that we will not find 70% of all the possible segmentation points and of those we find only half will actually be true segmentation points. This is unfortunately not good enough to be used in an application. The main issue is that we are unable to find certain segmentation points, in particular those that are similar in colors and textures. The issue is that our changepoint detection is unable to detect these small changes. This is not a bad thing since otherwise the false positives would be overwhelming. Instead we need to find other descriptors that distinguish these small changes better.

That said, we still found someone interesting correlations between the final segmentation and the real segmentation. We were in this thesis only interested in finding the segmentation points that optimized a certain distance measurement. We could have been a lot more restrictive in order to limit the amount of false positives. This would mean we would only get the values that we were sure to be real segmentation points. This could be used as a first step in order to find the more complicated segmentation points with another method.

Although we lack more extensive user segmentation data we can still compare our method with the human segmentation. The best human reached an edit distance of 105 which puts our average of 185 in new light. Sure these numbers are not fully comparable but it gives us at least some insurance that our algorithm still performance somewhat reasonable.

If the segmentation algorithm was a slight disappointment the indoor/outdoor classification method was a success. Reaching 97% accuracy is a good result even if it the algorithms might be overfitted. This could easily be implemented in a live setting with good results.

Throughout the thesis we used two different distance measurements, the edit distance and the Jaccard similarity. The results were somewhat different, the Jaccard similarity was more likely to predict a segmentation point at the cost of false positives, but none of the two distance measurements presented better results than the other. The edit distance does however provide more freedom with its three parameters compared to the zero of the Jaccard similarity.

8.1 Future work

Any future work would need to foremost focus on finding descriptors that better describes the scene. The indoor/outdoor segmentation described in this thesis is on of those more complicated descriptors. We believe that these kind of descriptors add a new dimension to the solution compared to just using various low-level features. Some relevant descriptors could be the distribution of explicit objects or perhaps a more complex classification system with classes like city, garden, living room etc.

Combine this with better optimization, a more complex changepoint detection and a better method to combine the different descriptors and you could potentially get a method that are similar to humans in performance. Of course you will need a lot more data in order to make a solid statical analysis of the performance.

References

[1] Christopher J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[2] Franklin C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques - SIGGRAPH '84*, volume 18, pages 207–212, New York, New York, USA, January 1984. ACM Press.

[3] David A. Forsyth and Jean Ponce. Computer Vision: A Modern Approach. March 2002.

[4] Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing (3rd Edition). February 2006.

[5] David Hinkley and Edna Schechtman. Conditional Bootstrap Methods in the Mean-Shift Model. *Biometrika*, 74(1):85, March 1987.

[6] Wonjun Kim, Jimin Park, and Changick Kim. A Novel Method for Efficient IndoorOutdoor Image Classification. *Journal of Signal Processing Systems*, 61(3):251–258, February 2010.

[7] Thomas Leung and Jitendra Malik. Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons. *International Journal of Computer Vision*, 43(1):29–44, June 2001.

[8] K. Levi and Y. Weiss. Learning object detection from a small number of examples: the importance of good features. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 2, pages 53–60. IEEE.

[9] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[10] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics.* The Regents of the University of California, 1967.

[11] S. Marčelja. Mathematical description of the responses of simple cortical cells*. *Journal of the Optical Society of America*, 70(11):1297, November 1980.

[12] J.R. Movellan. Tutorial on Gabor Filters. *MPLab Tutorials, Univ. of California, San Diego*, 2005.

[13] Aude Oliva and Antonio Torralba. GIST Descriptor.

[14] Aude Oliva and Antonio Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision*, 42(3):145–175, May 2001.

[15] E. S. PAGE. CONTINUOUS INSPECTION SCHEMES. *Biometrika*, 41(1-2):100–115, June 1954.

[16] Andrew Payne and Sameer Singh. Indoor vs. outdoor scene classification in digital photographs. *Pattern Recognition*, 38(10):1533–1545, October 2005.

[17] John C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. 1999.

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

[19] Jonathon Shlens. A tutorial on Principal Component Analysis.

[20] M. Szummer and R.W. Picard. Indoor-outdoor image classification. In *Proceedings 1998 IEEE International Workshop on Content-Based Access of Image and Video Database*, pages 42–51. IEEE Comput. Soc, January 1998.

[21] Wayne A. Taylor. Change-Point Analysis: A Powerful New Tool For Detecting Changes, 2000.

[22] Jianxin Wu and Jim M Rehg. CENTRIST: A Visual Descriptor for Scene Categorization. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1489–1501, December 2010.