Water segmentation and classification

Utrecht University

Department of Information and Computing Sciences

ICA-3827364

Author:
Pascal METTES
3827364

Supervisor:
Dr. Robby T. TAN

A thesis submitted for the degree of
Master of Science (M.Sc.), 45 ECTS

August, 2013

Abstract

In this work, a unified approach to water detection in videos is provided from both a discriminative and generative perspective. Although the automatic detection of water entails a wide range of applications, little attention has been given to solve this specific problem. Current literature generally treats the problem as a part of more general recognition tasks such as material recognition or dynamic texture recognition, without distinctively analysing and characterizing the visual properties of water. In order to compensate for this lack of information, a discriminative algorithm is presented here by introducing a hybrid descriptor based on the joint spatial and temporal behaviour of local water surfaces. Furthermore, this work provides a mathematical analysis and intuitive interpretation of linear latent variable modeling for dynamic texture classification. Based on the analysis, a set of improvements is proposed for the purpose of generative water detection specifically.

In addition, a novel water database is presented in this work, which goes beyond databases from related fields in terms of quantity and variety of both natural and man-made water scenes. Both perspectives are experimentally evaluated on this database and a subset of the DynTex database for the tasks of video segmentation and classification. The experimentation performed on these two tasks indicate the effectiveness of the introduced algorithms for discriminating water from other but related dynamic and static surfaces and objects, outperforming well-known algorithms from directly related fields. The algorithms and database presented here form a basis of the relatively unexplored problem of water detection and can lead to tackling new problems such as (near) real-time detection, large-scale detection in images, and detection with camera movement.

Acknowledgements

My gratitude goes out to Robby Tan, who has supervised me during this thesis project, as well as during the experimentation project on material classification and the literature review on water detection which preceded this thesis project. His supervision has not only helped to create the road from problem to possible solutions in this thesis project, but more importantly, I have learned invaluable lessons on how to walk such a road in general. His guidance has taught me when to think broad and when to think narrow, how to investigate something new, the merit of an invariant and robust method over an ad-hoc procedure, and how to formally present findings. I am especially grateful for being handed this specific topic, since it provided me the opportunity to directly apply these lessons. Thank you Robby for your supervision and guidance!

Verder wil ik ook graag mijn ouders, mijn grootouders, mijn zusje, mijn kleine neef, en Sandra bedanken voor hun hulp en steun. Zij hebben mij de volledige vrijheid gegeven om te doen wat ik graag wil en hebben mij geholpen waar ze konden. Bedankt! En Sandra, bij deze ook formeel bedankt voor het scouten van uitstekende locaties om te filmen.

"You can't trust water: Even a straight stick turns crooked in it."

- W.C. Fields

"Human nature is like water. It takes the shape of its container."

- Wallace Stevens

Contents

List of Figures

Chapter 1

Introduction

The task of water detection is seemingly effortless for human observers, which is hardly surprising given the biological importance of water. While recent studies have indeed shown that humans are experts at such tasks [50, 51], there is little empirical knowledge on how water can be optimally recognized. In other words, there is little knowledge on which visual properties make water a unique texture. Perhaps the most insightful piece of knowledge on this subject is provided in the work of Schwind [49], that indicated the importance of polarizing light reflected from water surfaces. The experiments performed by Schwind showed a profound interest of insects such as bugs and beetles in the horizontally polarized light of shiny water reflections.

This insight does however not solve the problem of detecting water in image frames, due to the fact that cameras and video recorders do not retain polarization information. Given that the problem of water localization in images and videos is still hardly problematic for human observers, there are clearly other spatio-temporal visual aspects that make water unique. Therefore, this work is concerned with analysing water in a spatio-temporal environment and with providing novel algorithms to localize water based solely on visual information. More concretely, the primary goal of this work is to create an algorithm that is able to classify each pixel in a video sequence based on whether the pixel is part of a body of water or not.

Besides gaining empirical knowledge on the visual properties of water surfaces in natural scenes, the automatic detection of water in videos also has a wide variety of practical applications. Exemplary applications include the discovery of water in (inter-planetary) exploration, dynamic background removal, and aerial video analysis e.g. for map making. Furthermore, there are multiple direct applications in robotics, for example the automatic cleaning of spills or the detection of puddles and canals in autonomous driving systems.

Traditionally, automatic water recognition is studied from two perspectives, namely as part of larger and more generic recognition systems, such as material recognition [24, 33, 50, 58, 59] and dynamic texture recognition [15, 17, 19, 20, 39, 43, 47, 48, 63, 64, 65], and as part of specialized or restricted environments, including autonomous driving systems [45] and maritime settings [52]. Most current works in material and dynamic texture recognition hypothesize that target classes can be discriminated using distributions of local image features, modeled video frames, or motion statistics. Although interesting results have been reported, the descriptors and statistics themselves are generic, while there is also little analysis on what makes each texture unique, as acknowledged e.g. in [19, 20]. On the other hand, water detection systems in autonomous driving systems and in maritime settings make explicit and non-generalizable assumptions about the environment, such as horizon location, camera height and orientation, and relative sky-to-water positioning. These assumptions render the methods incapable for water detection from a broad scope. As can be deduced from this short overview, the specific problem of water detection has not received a lot of attention in current literature. To the best of our knowledge, water detection has so far not been tackled on the scale and broad scope as presented in this work.

1.1 Thesis goals and contributions

Given the limitations of existing methods for the specific task of water detection, the problem of detecting water in video material is tackled from two angles in this work. The first perspective entails the analysis and descriptor generation of the local spatial and temporal behaviour of water. The second perspective entails the global modeling of videos using Linear Dynamical Systems, along with a mathematical analysis and proposed improvements for water detection. In other words, this thesis attempts to solve the problem of water detection from respectively a discriminative and generative perspective.

More specifically, the contributions of this thesis are as follows:

- The use of residual videos from RGB videos using temporal mode subtraction to increase invariance to weather conditions, inherent water colour, and background reflections.

- A segmentation and classification algorithm that segments the pixels in each video frame based on the presence or absence of water. To this end, a spatial and a temporal descriptor are introduced for direct local water identification. Each local descriptor in a video frame is used as a feature vector for probabilistic classification using Decision Forests [2, 12]. The resulting local probabilistic information is then subject to a spatio-temporal regularization step using Markov Random Fields [4] to yield a final segmentation for the frames in a video.

- A mathematical analysis and intuitive interpretation of Linear Dynamical Systems, a well-known approach to the modeling of videos. Based on this analysis, multiple direct limitations of LDS are addressed. Improvements proposed in this work include an optimization in the parameter learning stage, a novel distance measure between learned models, and a pyramid-structured formulation of LDS.

- Thorough experimental evaluation on a novel and publicly available database for the specific purpose of water segmentation and classification. Although databases containing water videos do exist (e.g. [44]), they lack the quantity and variability desired for thorough analysis and evaluation, since they are not created for the direct purpose of water detection. Therefore, another contribution of this work is the introduction of a novel database consisting of 260 videos of various water and non-water videos in a wide range of natural and man-made scenes. The water database, further discussed at the end of this Chapter, will be made publicly available to encourage further research into this topic. Experimentation is furthermore performed on a selected subset of the DynTex database [44].

- Comparison against a wide variety of related algorithms for the task of water video classification (i.e. classification of a video as either water or non-water).

For both the discriminative approach (item 2) and the generative approach (item 3), descriptors are extracted from train videos in order to learn which type of behaviour belongs to water or non-water. In the test stage, a video can be provided to the learned model. Based on the classification and possible regularization of the descriptors, each pixel in that video yields a binary label indicating the presence or absence of water. Therefore, the algorithms can be seen as a function which maps RGB values of video frames to binary labels.

The rest of the thesis is organized as follows. First, this Chapter is concluded with the details of the novel water database, while Chapter 2 further discusses the fields related to computational water recognition. Chapter 3 provides the analysis and descriptor generation of local water patches, along with probabilistic classification and global regularization; all part of the water segmentation algorithm. This is followed by Chapter 4 on the study and improvements of Linear Dynamical Systems, after which the algorithms are evaluated in Chapter 5. Lastly, the thesis is concluded in Chapter 6.

1.2 The water database

In order to tackle the challenge problem of water detection, experimental evaluation is performed in this work on a novel database. The water database contains a set of positive and negative videos (i.e. water and non-water videos), from a wide variety of natural scenes. In total, the database consists of 260 videos, where each video contains between 750 and 1500 frames, all with a frame size of 800×600. The positive class consists of 160 videos of predominantly 7 scenes; oceans, fountains, ponds, rivers, streams, canals, and lakes. The negative class on the other hand can be represented by any other scene. In the database, categories with seemingly similar spatial and temporal characteristics are chosen, including trees, fire, flags, clouds/steam, and vegetation. Multiple examples of the categories of the database are shown in Figure 1.2. Since the focus of this work is aimed at characterizing the behaviour of water, camera motion is considered a separate problem and it is therefore not integrated into the database. An overview of the classes and primary categories is shown in Figure 1.1.

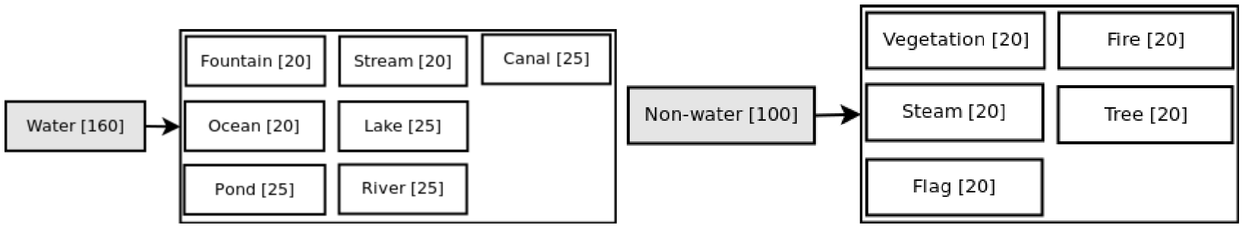
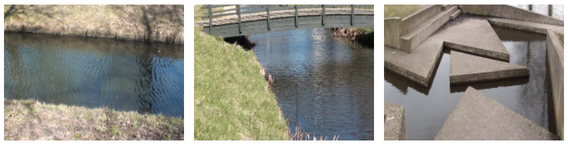
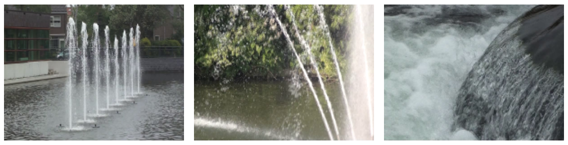


Figure 1.1: Overview of the water database.


(a) Canal.


(b) Fountain.

(c) Lake.

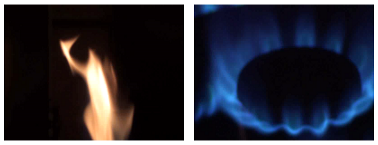
(d) Pond.

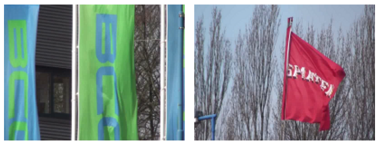
(e) Stream.

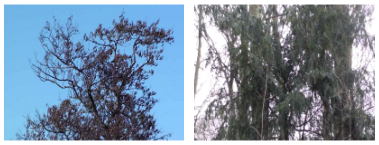
(f) River.

(g) Ocean.

(h) Clouds/steam.


(i) Fire.


(j) Flags.


(k) Trees.

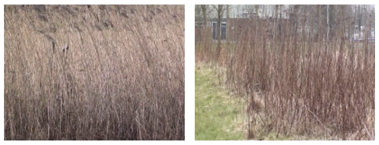

(l) Vegetation.

Figure 1.2: Example frames of videos in the novel water database.

Chapter 2

Related work

As stated in the Introduction, water detection in videos has mostly been tackled as a part of larger detection/recognition problems. Broadly speaking, water has formed a target class in the problems of static material classification, dynamic texture recognition/segmentation, and in large-scale recognition problems such as scene and event recognition. In this Chapter, these three lines of work are separately discussed.

2.1 Static material classification

Original work on material classification attempted to discriminate materials photographed in laboratory settings, e.g. based on distributions of Gabor filter responses [28, 58] and based on image patch exemplars [59]. More recently, the classification problem has shifted to the real-world domain with the introduction of the Flickr Materials Database [51], a database that includes water as one of the target classes. Current approaches attempt to discriminate materials based on concatenations of local image feature distributions [24, 33, 50]. Although it is possible to apply these approaches to the more specific problem of water recognition, they are severely restricted in multiple aspects. First, the algorithms are concerned with classification solely from images, meaning that any form of temporal information is neglected. More importantly, the algorithms provide a black-box process, where it is unknown how or why certain materials (such as water) can be discriminated using the recommended image features (e.g. SIFT, Jet, Colour, etc.).

2.2 Dynamic texture recognition and segmentation

A field more directly related to the problem tackled in this work is dynamic texture recognition. According to Nelson and Polana [40], dynamic textures are instances of a class of motions, common in natural environments, which exhibit structural or statistical self-similarity in both space and time. Typical examples include fire, trees, water ripples, flags, turbulent weather patterns, or a flock of birds. Note that the self-similarity is implicitly required to be non-trivial in this definition, i.e. a lack of movement is not considered temporal similarity.

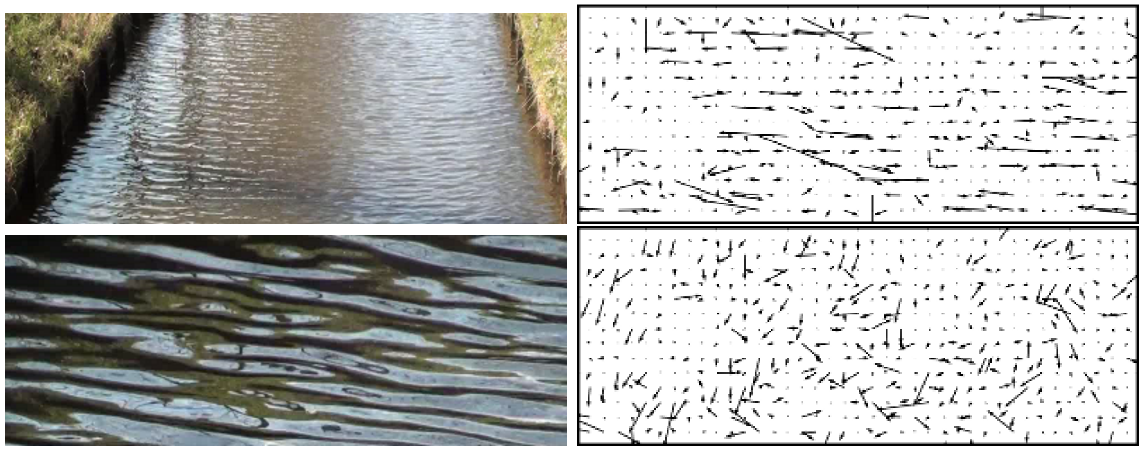


Figure 2.1: Two examples of water in a natural environment, along with an estimated flow field using resp. Lucas-Kanade and Horn-Schunck. In both cases, the dominant direction of the water ripples (resp. to the bottom and to the bottom-left) are not captured.

In dynamic texture recognition, there is a clear dominance of two approaches. The first approach is the discrimination of dynamic textures based on two-frame motion estimation. Generally, well-known conventional optical flow methods are used for dense motion estimation, after which classification is performed based on global motion statistics. Global statistics include the curl and divergences of the flow field, and the probability of having a characteristic flow direction and magnitude [19, 20, 43]. Although high recognition rates have been reported for such methods, the use of conventional optical flow is problematic for water surfaces in natural scenes. In general, water does not meet the conditions of optical flow, such as Lambertian surface reflectance, pure translational motion parallel to the image plane, and uniform illumination [1]. Initial investigations into optical flow-based water classification in this work have indicated that conventional flow methods cannot properly estimate water motion, as exemplified in Figure 2.1. Another pressing problem is that flow-based methods do not provide a way to the segmentation of water, only classification. Algorithms have been proposed for the segmentation of purely static and moving textures in a scene [18], but these do not generalize to the segmentation of water specifically, since other non-static textures are likely to co-occur with water in the same scene.

A second dominant approach is the global modeling of video frames using Linear Dynamical Systems (LDS) [8, 9, 15, 16, 39, 48]. In its essence, LDS is a linear latent variable model which projects video frames to a lower dimensional space and tracks the temporal behaviour of the elements in that lower dimensional space. The use of LDS in dynamic texture recognition was first popularized by the works of Saison et al. [48] and Doretto et al. [15], mostly due to the proposed relatively efficient parameter learning procedure and the encouraging classification results. The original work has later been extended for the purpose of segmenting pixels into different regions [16], mixtures of dynamic textures [9], or clustering texture videos [39].

Although the use of LDS is intuitively appealing, it is rather limited in its original form, where a single LDS model is learned for a whole video. As a result, the original LDS formulation cannot handle multiple objects/textures in a single video. Furthermore, it is claimed that LDS can capture the essence of a dynamic texture [15], but little investigation has been done as to what the learned essence actually entails. As listed above, multiple extensions have been made to handle the presence of multiple textures, but in current literature, LDS is used either as a segmentation or classification method, but not as a joint segmentation-classification problem (i.e. dividing the pixels into different coherent parts and classifying each part, as is done in this work by classifying each pixel as either water or non-water).

A third approach worth mentioning is the work on texture classification using (modified) Local Binary Pattern Histograms. A Local Binary Pattern (LBP) is an image descriptor which transforms a comparison of an individual pixel to a set of neighbouring pixels into a binary string [41]. Histograms are then formed by generating distributions of binary strings. In the context of static and dynamic textures, multiple extensions of LBP have been proposed for classification purposes, including but not limited to Dominant LBP [32], Completed LBP [22], and temporal extensions such as Volume LBP [65], LBP-TOP [63, 64], and ST-LBP [62]. So far however, LBP has pre-dominantly been used for global classification, not segmentation. In addition, the works on dynamic texture classification using a form of LBP provide little direct comparison to other works on dynamic texture classification [63, 65], making it hard to assess the strength of the proposed algorithms. Lastly, similar to the other works discussed in this Chapter, there has not been a specific investigation into whether LBP is useful of water detection specifically.

2.3 Scene and event detection

From the above discussion on material and dynamic texture recognition, it is evident that it is not possible to derive which visual properties make water unique. This trend is even more obvious in large-scale settings such as scene recognition and event detection. In these problems, the detection of water can be quite informative, since it provides a clear indication of a set of possible scenes or actions. Similar to the works on static material classification, the classes in large-scale classification and recognition tasks are predominantly tackled from a purely spatial perspective. For example in the work of Bosch et al., categories including water are discovered in an unsupervised manner using probabilistic Latent Semantic Analysis (pLSA) [3]. On the other hand, in the work of Li and Fei-Fei [29], distributions of SIFT descriptors are used to classify scenes e.g. based on the presence of water. Furthermore, based on generic texture classifiers, it is also possible to create a high-level image representation in terms of objects and textures, again with water as a possible category [30].

Although the generic spatial algorithms in a large-scale setting are arguably limited for water detection specifically, it must be noted that these works do not have the same primary objective as this work. In scene and event detection, an approximate detection of water is sufficient, since their main motivation lies in detecting higher-level interpretations, not accurate detection. Also, the computational effort for detection purposes are kept to a minimum, given the large amount of possible objects and textures within a single scene.

Chapter 3

Local spatial and temporal classification

Throughout this thesis, two perspectives are provided to the water detection problem. In this Chapter, a local discriminative perspective is provided, where both the spatial and temporal behaviour of patches of video frames are used to analyse water. A generalized overview of the whole method introduced here is shown in Figure 3.1. The layout of the Chapter is as follows. First, the process of temporal mode subtraction, which yields residual videos, is explained and justified. After that, the local temporal behaviour of water is analysed and the temporal water descriptor distilled from that analysis is discussed. The same process is redone for the local spatial behaviour and corresponding spatial water descriptor. Given the computable descriptors, the process of local direct probabilistic classification using Decision Forests is explained. The Chapter is concluded with the final step of the algorithm, namely the use of spatio-temporal Markov Random Fields as a regularization step for the creation of coherent frame-to-frame segmentations.

3.1 Temporal mode subtraction

A main concern with the identification of water surfaces on a local scale is the inherent variability they possess. The variability is subject to multiple conditions, including water colour, different water ripples based on wind conditions, general weather conditions, and possible object reflections. One possible approach is to exploit consistencies within that variability for water detection, e.g. exploiting the possible presence of blue skies, white clouds, and brown trees. This approach however directly limits the scope and possible applications of the algorithm. For that reason, all videos in the database are pre-processed in an off-line step (i.e. performed once outside the scope of the water segmentation algorithm) in order to increase invariance to multiple elements of variability.

The main angle behind this pre-processing step is a shift of focus to water ripples. To achieve this shift of focus, the water colour and/or background reflections of each pixel in a video are first obtained, after which they are removed, leaving residual images. In the context of water, these residuals represent the water ripples, since the ripples are the primary cause of the disruption of the reflection. Although the use of residual videos over original videos

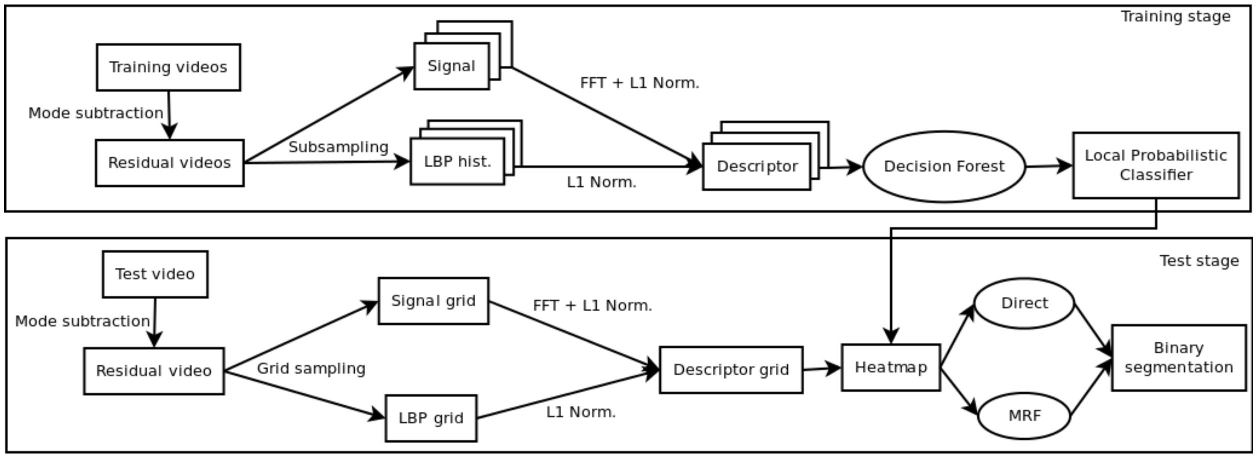


Figure 3.1: An overview of the water segmentation method. Note that the test stage is performed for all frames of the test videos.

creates an invariance to water colour and background reflections, it is unknown which colour values actually represent the reflections. Fortunately, upon observation of water in natural scenes, it is clear that water ripples only cause a temporary disruption of the dominating reflections. The most often occurring values at each spatial location location over time is therefore assumed to represent the reflections.

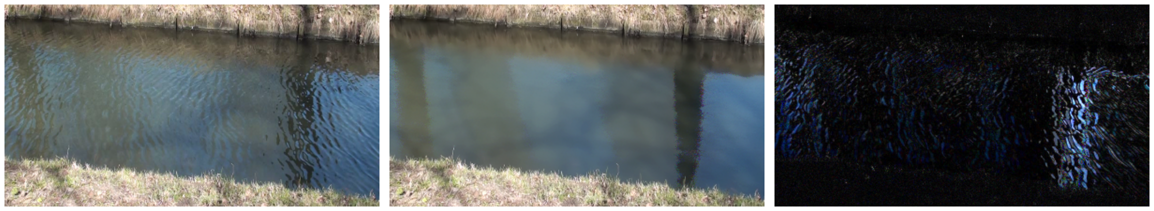
Given this observation, a video sequence $\{I_t\}_{t=1}^{f}$ of f frames can generate a residual sequence $\{R_t\}_{t=1}^{f}$ for each pixel of each frame using the following simple equation:

$$R_t(x,y)[c] = |I_t(x,y)[c] - M(x,y)[c]|, \tag{3.1}$$

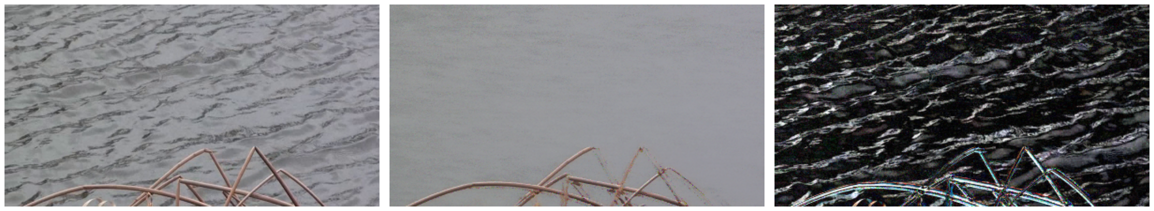
for each $c \in \{R,G,B\}$, where $M(x,y)$ denotes the temporal mode of the original video at pixel (x,y). Contrary to the mode statistic, which by definition finds to most often occurring element in a list, other statistics such as the mean and median are inappropriate choices here, since these statistics blend ripple and background information. A typical frame, temporal mode, and corresponding residual frame of two videos are shown in Figure 3.2. This simple off-line process can successfully find the most dominant elements of reflection, such that the residual frames contain the water ripples.

3.2 Local temporal water descriptor

Given the residual videos, the local behaviour of water is investigated from a purely temporal and spatial point of view. The idea behind the temporal water descriptor is that the water ripples indicate the motion characteristics and are hypothesized to constitute several aspects. Most notably, it is hypothesized here that the motion of water ripples is gradual and that the motion is repetitive since ripples re-occur at the same location over time. The characterization of water motion as both gradual and repetitive is intuitively sound and furthermore consistent with the knowledge of the physical properties of water. For example, it



(a)



(b)

Figure 3.2: Two examples of water surfaces, along with their temporal mode, and resulting residual frame. The effects of the residual frame are enhanced for visualization purposes.

is well known that water waves are forced (and therefore bounded) by gravity and surface tension. More concretely, the dispersion relation of water states that there is a dependence between the wavelength and phase speed of waves [54, 56], i.e.:

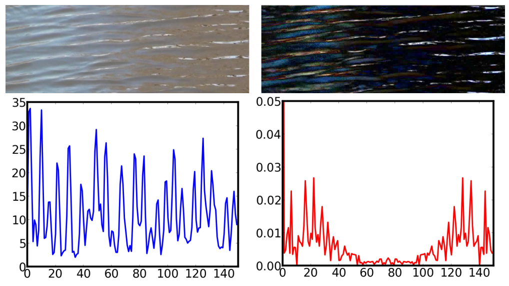
$$\omega^2 = gk, \tag{3.2}$$

where ω denotes the angular frequency (radians per second), and g and k denote resp. the gravitational constant and the wavenumber (radians per metre). The above equation is in actuality a special case of the dispersion relation where the depth D to the bottom can be ignored (e.g. in the case of oceans). More generally, a lower depth value has an additional retarding affect on the waves:

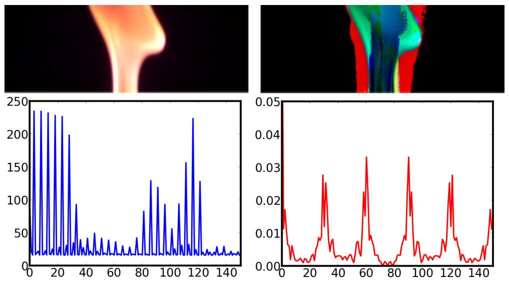
$$\omega^2 = gk \tanh(kD). \tag{3.3}$$

As a result, due to the inherent nature of water waves, it is hypothesized that the luminance values of water surfaces on a local scale change gradually from low to high values and vice versa. As a result, the temporal behaviour is expected to represent a pseudo-sinusoidal pattern, given that similar waves enter and exit a local scene in a constant fashion. Other dynamic and static textures/objects - such as fire, flags, trees, or static objects - might partially share the local temporal motion properties of water, but not statistically to the same extent.

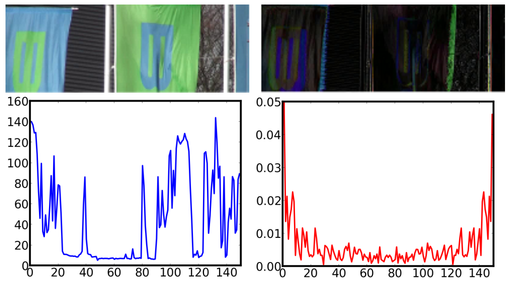
Based on the defined hypotheses, the next step is to create a descriptor that judges a local spatio-temporal patch with respect to these hypotheses. To achieve this, an m-dimensional signal is first introduced, which represents the mean brightness value of an $n \times n$ image patch for m consecutive frames at exactly the same location. In other words, given a pixel centre (x, y) and m consecutive frames, the mean brightness value of an $n \times n$ patch around the centre pixel is computed for those m frames. The result is a list of brightness values



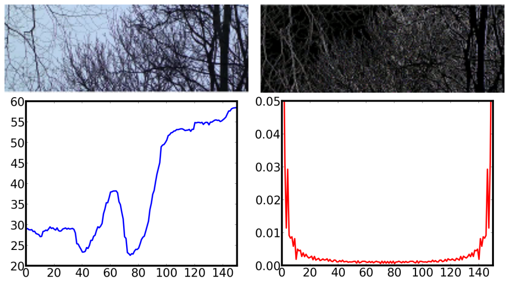
(a) Water.



(b) Fire.



(c) Flag.



(d) Tree.

Figure 3.3: Multiple video examples, along with their residual, temporal signal, and normalized FFT of the signal.

that can be seen as a 1-dimensional signal. Figure 3.3 shows multiple examples of water and non-water scenes, along with a mean brightness signal of a selected location. From the Figure, the initial thought regarding regularity and repetition are already clearly visible; the exemplary signals of the water scenes change more gradually between two extremes and are also more repetitive.

Given the m-dimensional signals, an immediate thought is to use them directly as the descriptor. The direct use of the signal itself as the descriptor is however erroneous, due to lack of invariance. Signals with similar levels of smoothness and regularity can have a large distance when comparing the signals directly. A descriptor based on the m-dimensional signal should in effect be invariant to temporal shifts, brightness shifts, and brightness amplitudes. Rather than creating a distance measure that explicitly enforces these types of invariance by means of normalization and distance recalculation for all possible temporal and brightness shifts, a descriptor is created here by extracting the signal characteristics using the 1-dimensional Fourier Transform. For an m-dimensional signal S, the corresponding Fourier Transform F is also m-dimensional, where each component $i \in \{1, m\}$ is computed as:

$$F_i = |\sum_{j=1}^{m} S_j e^{-2\pi i j \sqrt{-1}/m}|. \tag{3.4}$$

In other words, from the signal, an m-dimensional descriptor $[F_1, .., F_m]$ is computed. Although this descriptor is invariant to the temporal and brightness shifts, it is not invariant to brightness amplitudes. Therefore, the final temporal descriptor is generated by performing

$L1$-normalization on the components, such that two signals are compared by the distribution of the Fourier Transform, i.e.:

$$F_i = \frac{|\sum_{j=1}^{m} S_j e^{-2\pi ij\sqrt{-1}/m}|}{\sum_{k=1}^{m} |\sum_{l=1}^{m} S_l e^{-2\pi kl\sqrt{-1}/m}|}. \tag{3.5}$$

The general use of the Fourier Transform to analyse and detect water has already been pointed out in Tessendorf's paper on simulating ocean water [56]. In that work, it is shown in the form of a simple experiment that the spatio-temporal (i.e. 3D) Fourier Transform is useful for the modeling of water waves. Since the spatial part of that experiment is based on the height of each part of the water, it is not applicable to video data, where such information is not given. The temporal part in the form of Eq. 3.5 can still hold merit.

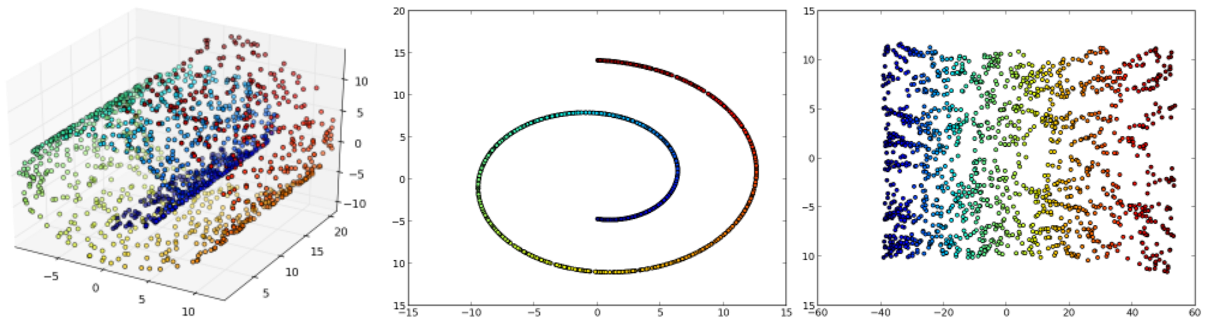
The choice of the signal length m constitutes a trade-off. Ideally for classification purposes, m is as large as possible. This is due to the fact that the longer the signal, the less likely it is that non-water surfaces can mimic the temporal behaviour of water. From a practical point of view however, m is as small as possible, such that temporal discontinuities and camera shifts do not form a problem. Since the focus of this thesis lies primarily on discrimination abilities, while still being able to detect obvious discontinuities, m is mostly set to 200 frames in this work. The value of the patch size n is set empirically to 11 pixels in width and height.

3.2.1 Visualizing separation using non-linear projections

Although the above explanation and Figures provide an intuitive justification of the ability of the descriptor to discriminate local water signals from non-water signals, it is unknown how well the descriptor can actually generate any form of separation in the m-dimensional feature space. Ideally, we would like to inspect the feature space directly to see whether local water and non-water patches are grouped together. Since the feature space is high-dimensional (typically 150-dimensional or more), it is impossible to directly inspect the space, and therefore it is required to project the feature space to 2 dimensions. Note that within the classification framework, the original high-dimensional space is utilized for the discrimination of local descriptors. The use of the projection is primarily for visualization purposes. If clear patterns in a projection (i.e. dimensionality reduction) of the feature space can be seen, there is a good promise for the discrimination powers of the used descriptor in the original space.

A classical way to reduce the dimensionality of the feature space is to use Principal Component Analysis (PCA) and drop all but the highest eigenvalues. However, methods such as PCA are unable to handle non-linear structures within the dataset. For that reason, the Isomap algorithm by Tenebaum et al. is used here [55]. The Isomap algorithm performs non-linear dimensionality reduction by computing local metric information of the elements in the dataset to learn the underlying global geometry of the whole dataset.

The Isomap algorithm can be decomposed into three steps. In the first step, it is determined which data points are neighbours in the feature space based on the distances between



(a) Swiss roll. (b) Side view of the roll. (c) Isomap projection.

Figure 3.4: Illustration of the workings of Isomap. Best viewed in colour.

pairs of data points. This can be done either by computing the distance to all data points within a distance of ϵ or the K nearest neighbours for each data point. The set of computed distances can then be seen as a weighted graph G, where the edges between neighbouring points have a weight proportional to their computed distance. Given a graph G, the second step of the algorithm is aimed at computing the geodesic distances between all pairs of points by computing their shortest distance patch distances in G. The third and final step constructs an embedding of the data in the specified dimensional space that best preserves the manifold's estimated geometry.

An illustrative example of the prowess of Isomap is the problem of dimensionality reduction of a swiss roll. The 3D feature space of the swiss roll and a side view of the feature space are shown in resp. Figure 3.4a and Figure 3.4b, where the pattern of outwards movement is obvious to human observers. If the geometric embedding is however not taken into account, data points in the blue region could have a smaller distance to points in the red region than points in the green region. As can be seen in Figure 3.4c, the Isomap algorithm can handle the problem of finding a suitable 2D projection of the swiss roll.

Equipped with the Isomap algorithm, it is possible to project the high dimensional normalized Fourier descriptors to 2 dimensions. To inspect the projections for separation, a number of local water and non-water patches are randomly selected from a range of videos.

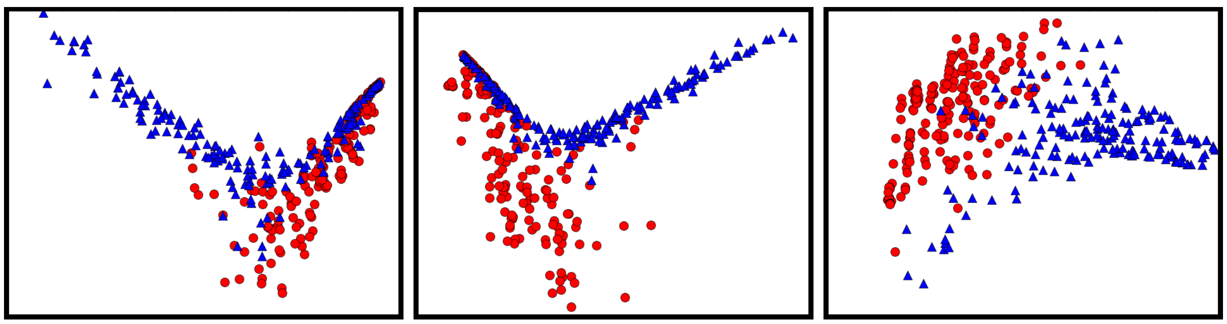


Figure 3.5: Isomap projections of resp. the original signals, the FFT of the signals, and the normalized FFT of the signals for a selection of patches..

In Figure 3.5, the projected feature spaces of the original signals, the Fourier descriptors, and normalized Fourier descriptors are visualized for a number of local water patches against a number of local tree patches. From the projections it is clearly visible that the original signals and FFT descriptors are unable to directly separate water and non-water patches. The normalized FFT descriptor is able to create a form of separation, although not a linear separation in a 2D projection of the feature space.

3.3 Local spatial water descriptor

The above defined descriptor captures the temporal behaviour of water, but ignores the local spatial information, most notably the spatial layout of water waves and ripples. In general Computer Vision literature, a great number of spatial local image features and descriptors have been introduced (see e.g. [38] for an overview). Given that water waves, ripples, and fountains are highly deformable, a descriptor is desired which provides spatial information on a local patch without requiring an explicit model of water waves. To meet this desire, the local spatial characteristics of water surfaces are extracted using Local Binary Pattern histograms [41].

For a single pixel, the Local Binary Pattern is computed by comparing the grayscale values of the pixel to a set of local spatial neighbours. The set of neighbours are usually selected in a circularly symmetric way, with a defined number of neighbouring pixels to compare to (P) and the radius of the circle (R). Figure 3.6 provides a number of examples for different values of P and R, where the pixel closest to the black dot is used as the compared pixel.

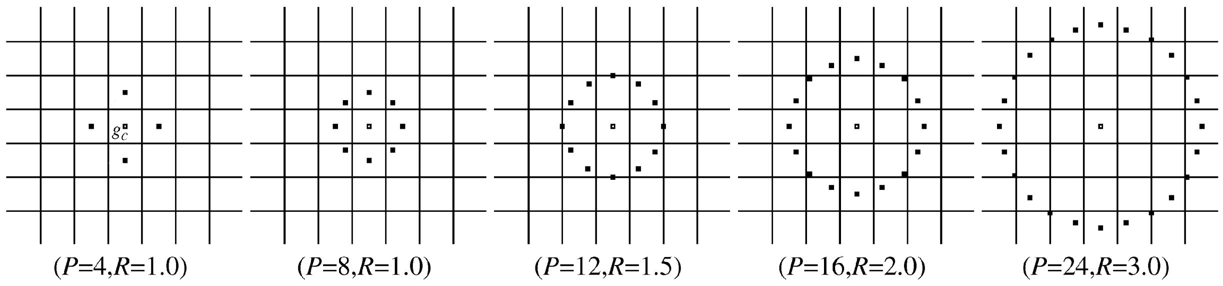


Figure 3.6: Set of circular neighbour sets for possible LBPs. Image courtesy of [41].

In this work, the 8 direct neighbours of a pixel are used for comparison, i.e. P and R are set to resp. 8 and 1. As such, the Local Binary Pattern value of a single pixel is computed as:

$$LBP_{8,1}(g^c) = \sum_{p=0}^{7} H(g_p^c - g^c)2^p, \tag{3.6}$$

where g^c denotes the centre pixel for which the LBP value is computed, $\{g_i^c\}_{i=0}^{7}$ denotes the set of direct neighbours of g^c, and $H(\cdot)$ is the well-known discrete Heaviside step function, defined as:

$$H(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0. \end{cases} \tag{3.7}$$

In other words, Eq. 3.6 states that a pixel is compared to a set of neighbouring pixels, where each comparison is reduced to a binary value (1 if the neighbour is higher or equal, 0 otherwise). The result is a binary pattern, that represents an integer in a decimal system. Since 8 neighbours are used in the comparison, the corresponding LBP value can take $2^8 = 256$ values. In order to compute a LBP histogram of a local patch, the LBP values of the pixels in the patch are computed and placed in their corresponding integer bins of the 256-dimensional histogram. The resulting histogram is normalized afterwards. Similar to the temporal descriptor, the patch size is set empirically, in this case to a value of 27 pixels in width and height. For both the temporal and spatial descriptors, two feature vectors are compared using the L_2 distance.

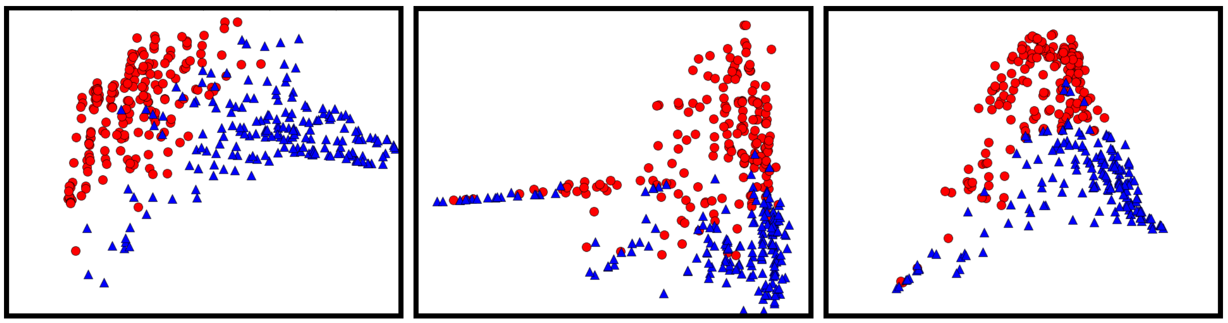
The local spatial LBP descriptor and local temporal normalized Fourier signal descriptor describe different elements of local video parts, but can be combined to generate a hybrid water descriptor. For a pixel (x, y) at frame f, the LBP histogram can be computed based on a patch around (x, y) within the same frame, while the temporal descriptor can be computed based on a patch around (x, y) for frame f and its $t - 1$ consecutive frames.

3.3.1 Relation to literature on Local Binary Patterns

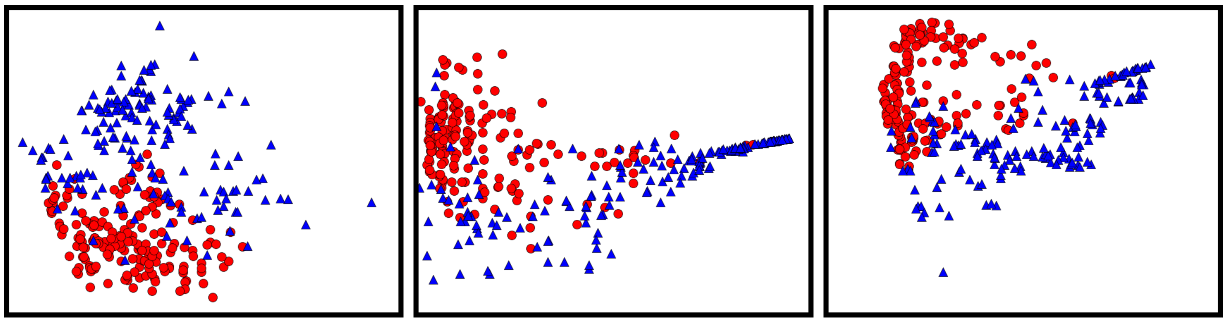
As stated above, a primary justification for the use of Local Binary Pattern histograms as a spatial descriptor is due to the pseudo-orderless nature of the descriptor, meaning that water waves do not need to be modeled explicitly. In addition, the high dimensionality of the histograms provide desirable discrimination abilities. Similar to the temporal descriptor, the practical validity of the LBP histograms can be shown by examining the projected feature space. As shown in Figure 3.7, the LBP histograms of water and non-water regions are rather split, albeit with a certain level of overlap. The early fusion (feature vector concatenation) [53] of the descriptors into a hybrid descriptor however, results in a feature space where its projection is almost nearly linearly separable for the randomly selected local patches.

The use of a pseudo-orderless descriptor was not deemed important in the early stages of the investigation into the spatial behaviour of water. This additional requirement came to light after the investigations into explicit water motion modeling turned out to be impractical. A notable example is the investigation into the modeling of water using Gabor filters. Since well-known anisotropic Gabor filters such as edge of bar filters to some extent resemble water waves, an intuitive approach is to attempt to match these filters at multiple scales and orientations to local water image parts. In practice however, water waves, ripples, and fountains are far more complex, such that a match is nearly always impossible to find.

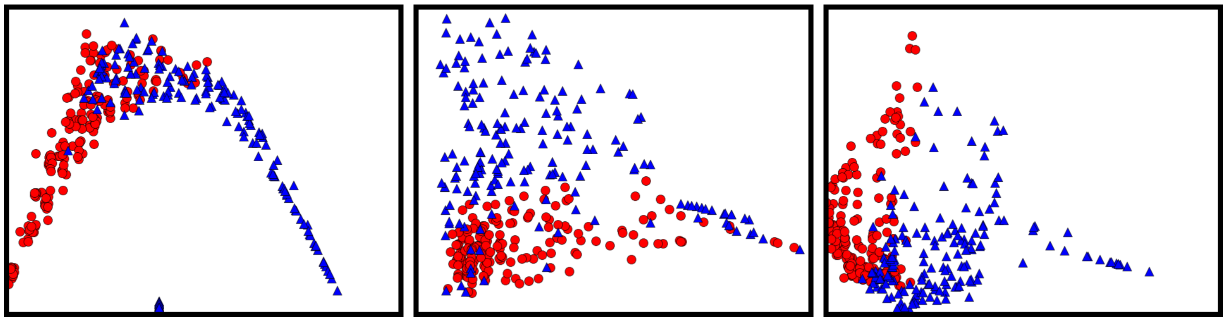
The step to LBP for local spatial description is consistent with related literature on dynamic texture recognition. Multiple related works have attempted to perform classification of dynamic textures based on LBP histograms. For example, Zhao and Pietkäinen have



(a) Water (blue) vs. Tree (red).



(b) Water (blue) vs. Cloud (red).



(c) Water (blue) vs. Flag (red).

Figure 3.7: Isomap projections for local water patches verses resp. local tree, cloud, and flag patches. In each example, the projections from left to right are for the temporal, spatial, and hybrid descriptor.

proposed on multiple occasions the use of Volume Local Binary Patterns - the natural spatio-temporal extension of LBP which also compares to temporal neighbours - for dynamic texture classification [63, 64, 65]. In overlapping work, Zhao and Pietkäinen have also introduced LBP-TOP to deal with the descriptor length of VLBP [64, 65]. In LBP-TOP, the LBP histograms are computed for the Three Orthogonal Planes of a pixel (the $xy-$, $xt-$, and $yt-$ planes), which reduces the descriptor length considerably compared to VLBP, but is still 3 times larger as standard LBP. Furthermore, in independent recent work, Ren et al. have proposed enhanced LBP features for dynamic texture recognition [47].

Even with the promising classification results, VLBP is impractical for local direct identification, since each histogram would have a length of 2^{14} or even 2^{26}, due to the fact that both spatial and temporal neighbours need to be compared. Similar statements can be made regarding LBP-TOP. Therefore, the original purely spatial LBP descriptor is used here.

3.4 Probabilistic classification using Decision Forests

Now that the behaviour of water has been defined on a local temporal and spatial level, the next step is to exploit the descriptors for probabilistic classification. Rather than computing distributions of descriptors as is usual in global classification tasks, the descriptors are used directly as feature vectors for probabilistic classification using Decision Forests [2, 12].

In the train stage of the algorithm, signals and LBP histograms are extracted from local patches of train videos and added to the list of feature vectors for the Decision Forest. Since the number of patches per frame can be considerably large, selecting patches from a uniform grid for each frame of each train videos is highly undesirable, given the amount of time required for training. Therefore, a random sampling approach is employed here by selecting a small number of patches from random locations for each frame of each train video. With this approach, roughly 200.000 to 350.000 descriptors are extracted from the train videos and used to learn the Forest, depending on the amount of patches to select per frame.

With a learned Forest, the local patches of the test videos can be classified. Contrary to the train stage, descriptors need to be computed in the test stage from all parts of the frames. Since computing the descriptors for each pixel is too expensive to ensure a completion in a reasonable amount of time, patches are extracted from a dense rectangular grid with a pixel stride strictly larger than 1. In practice, the stride is between 11 and 21 pixels, based on the used descriptor. Each locally extracted descriptor is individually classified using the learned Decision Forest, yielding a matrix of probability outputs. This matrix of probabilities, which can be visualized as a heatmap, serves as input for the final stage of the algorithm, where the segmentation for each pixel is computed by regularizing and interpolating the heatmap values.

For sake of completeness and mostly in order to be self-contained, a general overview of the theory of Decision Forests is provided here. Decision Forests combine the ideas of decision trees, such as the Classification And Regression Tree (CART) [6], and ensemble methods, which have shown that combining generic (weak) classifiers yields greater accuracy and gen-

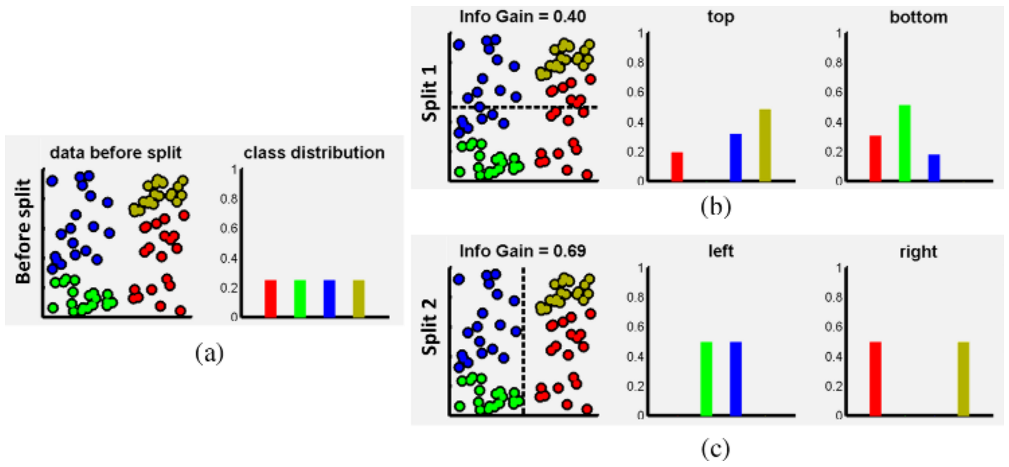


Figure 3.8: Two possible split functions their information gains.

eralization. The Decision Forests used in this work are based on the unified, efficient model of Random Decision Forests by Criminisi et al. [12], as defined by the ALGLIB library [2].

In a classification environment, a decision tree is a decomposition of the feature space into increasingly smaller subspaces. In other words, the feature space in which the train objects reside is repeatedly split. The goal of a decision tree is to find the splits such as to optimally discriminate between different target classes. At a node j, the respective train subset S_j is "best" split into S_j^L and S_j^R based on maximizing an objective function at that node:

$$\theta_j^* = \arg\max_{\theta_j \in \mathcal{T}} I_j, \tag{3.8}$$

where $\mathcal{T}$ denotes the set of random splits, I_j is the information gain resulting from split j, and S_j^L and S_j^R denote the non-overlapping split of S_j. I.e:

$$\begin{aligned} S_j^L &= \{(\mathbf{v},\mathbf{y}) \in S_j | h(\mathbf{v},\theta_j) = 0)\}, \\ S_j^R &= \{(\mathbf{v},\mathbf{y}) \in S_j | h(\mathbf{v},\theta_j) = 1)\}. \end{aligned} \tag{3.9}$$

In Eq. 3.9, $h(\mathbf{v},\theta_j)$ is the test function of the split node. Ideally, the data is linearly separable, which leads to finding a split function that completely isolates target classes. Since this is rarely the case in practice, the split function that yields the largest information gain is chosen. Figure 3.8 provides a visual example of two possible splits and their information gains. It is clear that the second split is more desirable, since it isolates the blue and green classes from the yellow and red classes (and vice versa).

More formally, the information gain of a split function is defined using the Shannon entropy:

$$H(S) = -\sum_{c \in C} p(c) \log(p(c)), \tag{3.10}$$

where S is the set of data points and C is the set of target classes. The less uniform the distributions of the child nodes are, the higher the information gain. This is measured using the following equation:

$$I = H(S) - \sum_{i \in \{L,S\}} \frac{|S^i|}{|S|} H(S^i). \tag{3.11}$$

The process of repeatedly choosing a split function that maximizes the information gain in a node results in a decision tree. The target class of a test descriptor can be classified using a decision tree by going through the tree, based on the split function at each internal node, until a leaf node is reached. The target class is then predicted based on the distribution of data points within the leaf. The set of possible splits $\mathcal{T}$ can be extremely large for large dimensional problems. For that reason, a randomness parameter ρ is introduced. This parameter controls the degree of randomness in a tree. E.g. for $\rho = |\mathcal{T}|$, all possible splits are used, while for $\rho = 1$, only a single, randomly chosen, split is used.

Although the decision tree as defined above could be used directly for the classification in this framework, it is beneficial to use an ensemble of trees, i.e. a Decision Forest, for classification. A Decision Forest is a collection of randomly trained decision trees, where the key aspect of the model is the fact that its component trees are all randomly different from one another. In the Forest model, the individual trees are trained separately. However, due to the randomness value ρ and because different different trees can have different split functions (e.g. axis-aligned split lines, general oriented split lines, or quadratic split lines), a descriptor can be defined by different probability distributions over the set of target classes. Given T trees, the probability distribution over the set of target classes c of an input descriptor $\mathbf{d}$ is defined as the averaged probability over the trees:

$$P(c|\mathbf{d}) = \frac{1}{T} \sum_{t=1}^{T} p_t(c|\mathbf{d}). \tag{3.12}$$

In the context of supervised classification, the main parameters of the Forest need to be determined empirically. Criminisi et al. note that there are 6 main parameters which influence the behaviour of the Decision Forest the most: the maximum allowed tree depth D, the amount of randomness ρ, the number of trees T , the choice of weak learner model(s), the training objective function, and the choice of features in practical applications [12]. Within the ALGLIB library, not all parameters can be tuned. For example, the trees are always fully grown and never pruned [2]. The parameters T an ρ can be set actively and in the context of this algorithm, these parameters are set to a static value. It is possible to determine the optimal values for these parameters by means of (cross-)validation, but this is undesirable given the large amount of feature vectors on which the Forest needs to be learned.

3.5 Spatio-temporal regularization

In the previous step of the algorithm, it could also have been possible to use non-probabilistic decision classifiers such as Support Vector Machines [11] or k nearest neighbours. Also, it is possible to convert the probabilistic outputs of the Decision Forest directly to 1-0 decisions (i.e. water vs. non-water decisions), yielding a direct segmentation of the frames. The additional information on the probabilistic (un)certainty of classified local patches however opens the possibility to discrete optimization on the heatmaps. The discrete optimization takes the form in this work of a Markov Random Field [4, 31], which serves as a regularization step. The main reason for the additional regularization is that even for a very successful set of descriptors, a learned Decision Forest is bound to make miss-classifications on a large set of individually classified local patches.

The Markov Random Field attempts to regularize the probability values in the heatmap by enforcing consistency within neighbouring nodes (i.e. elements of the heatmap). More formally, the optimization problem of the MRF can be stated as a minimization problem with the following objective function:

$$f(x) = \sum_{p \in V} V_p(x_p) + \lambda \sum_{(p,q) \in C} V_{pq}(x_p, x_q), \tag{3.13}$$

with V the elements of the heatmap, and C the set of all cliques. The first term of the objective function - the data term - is then defined as:

$$V_p(x_p) = \begin{cases} 1 - M_p & \text{if } x_p \text{ is water} \\ M_p & \text{otherwise,} \end{cases} \tag{3.14}$$

where M_p denotes the probability of node (i.e. heatmap pixel) p of begin water. The second term - the prior term - is defined as the Potts model [5], which penalizes different labels within cliques:

$$V_{pq}(x_p, x_q) = \begin{cases} 1 & \text{if } x_p \neq x_q \\ 0 & \text{if } x_p = x_q. \end{cases} \tag{3.15}$$

Since this work is focused on binary labeling, Eq. 3.15 can be rewritten as:

$$V_{pq}(x_p, x_q) = |x_p - x_q|, \tag{3.16}$$

given that the label water is defined as 1 and the label non-water is defined as 0.

An interesting element within the MRF formulation are the cliques. Since a heatmap is computed for each frame of a test video, a natural notion is to perform regularization on each heatmap, as is shown in Figure 3.9b. This approach enforces spatial label similarity, which intuitively states that it is not expected that water and non-water regions are very small and that regions occur on a high number of occasions separately within a frame. However, a form of temporal regularity is also desired, since it is most certainly not expected

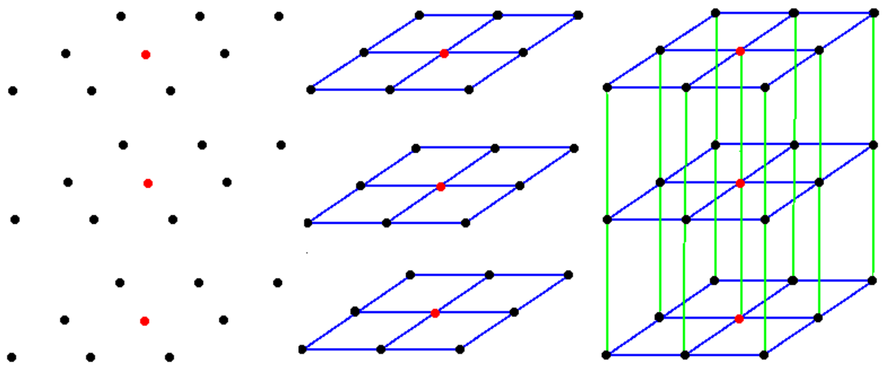

(a) None. (b) Spatial. (c) Spatio-temporal.

Figure 3.9: Three examples of possible dependencies, resp. no relations, spatial dependencies, and spatio-temporal dependencies. The red dots denote the centre pixel in each of the three frames used here.

that the same region switches rapidly between being water and non-water. Therefore, a spatio-temporal Markov Random Field formulation is opted here, as shown in Figure 3.9c.

Given the probabilities from the Decision Forest and the pair-wise spatial and temporal dependencies, the objective function of Eq. 3.13 can be minimized. The minimization procedure attempts to find a trade-off between the cost of deviating from the probability value of the Decision Forest and deviating from a consistent labeling of pair-wise neighbour. In order to find an approximate minimization of the objective function, the min-cut/max-flow algorithm of Boykov and Kolmogorov has been used [4].

The min-cut/max-flow algorithm of Boykov and Kolmogorov is an improvement of the standard augmenting path techniques on graphs. Shortly stated, standard augmenting path-based algorithms push flow from one target class (source) to another (sink), along non-saturated paths until a maximum flow is reached [14]. In their work on a comparison of min-cut/max-flow methods, Boykov and Kolmogorov found multiple empirical improvements, including the build of a search from sink to source, and the reuse of of search trees rather than building new trees from scratch. More information on the algorithm can be found in [4]. Since the regularization performed in this work is also a binary problem (water or non-water), the algorithm can be directly applied to this problem be assigning each target class to the source or sink.

An important element in the minimization procedure is the relative weight of the probabilities (data term) and spatial consistency (prior term), denoted by λ in Eq. 3.13. For a low value for λ, the individual probabilities are deemed important, resulting in a segmentation with a lot of detail, but also with outliers. On the other hand, a high value for λ results in a segmentation with little outliers, at the cost of loss of detail at borders between water and non-water regions.

Since not all pixels on a single frame are classified, the binarized heatmap only contains the segmentation result for a subset of the pixels on a rectangular grid. The segmentation results are therefore bi-cubicly interpolated such that each pixel is classified as either being water or non-water. A graphical overview of the last two stages of the algorithm are shown in Figure 3.10.

In Chapter 5, the whole algorithm explained in this Chapter is experimentally evaluated on the novel water database. This evaluation includes the empirical success of the individual and hybrid descriptors, the influence of regularization, and the influence of the λ term. On a last note, in recent independent work by Kontschieder et al. [25], the prediction step using Decision Forests and the regularization step using Markov Random Fields have been combined in a single algorithm. Although, the of their Geodesic Forest algorithm can beneficial for this algorithm, its use falls however outside the scope of this work.

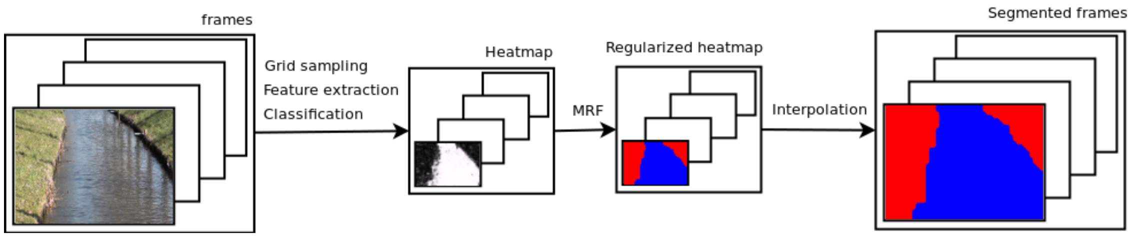


Figure 3.10: The steps of the algorithm for a single example.

Chapter 4

Latent variable modeling

A dominant approach in the field of dynamic texture classification is the use of Linear Dynamical Systems (LDS) [8, 9, 15, 16, 39, 48]. In its essence, LDS is a latent variable model which projects high-dimensional video frames to a lower dimensional latent space and tracks the temporal behaviour of the lower dimensional embedding. LDS has been opted on numerous occasions in the context of dynamic textures, investigations into the use of LDS for water classification and segmentation employed in this work have indicated multiple fundamental problems with the original formulation of LDS [15, 48]. Furthermore, little explanation has been given as to what the components of LDS actually entail and why the system works. For these reasons, this Chapter attempts to provide a mathematical explanation and intuitive interpretation of the elements of LDS to dynamic texture / water recognition, both for synthesis and classification. Based on a sub-optimal algorithm for parameter learning, the theoretical and empirical drawbacks and limitations of LDS are presented, along with proposed improvements for the purpose of water detection in video. The improvements include an optimization in the parameter learning stage, a novel distance measure between learned LDS models, and a pyramid structure for joint segmentation and classification. LDS is primarily interesting in the context of water detection, since - contrary to the purely discriminative algorithm of Chapter 3 - LDS is a generative process, which could mean that it can model the specifics of water, that could then be further exploited for discrimination.

4.1 LDS theory and parameter learning

For video sequences, a Linear Dynamical System is a stochastic dynamical model which attempts to capture the "essence" of the video, by modeling the spatial and temporal layout of the video. As such, LDS provides a global characterization that, due to its generative nature, can be used for tasks such as synthesis/reconstruction and classification. The first assumption made in the context of LDS and dynamic textures is that sequences of images containing a dynamic texture are realizations of second-order stationary stochastic processes [48]. This assumption has been conveniently chosen given that from the field of System

Identification, it is known that such a process can be modeled as the output of a LDS driven by white, zero-mean Gaussian noise [35]. In the next section, it is empirically verified whether this assumption is a realistic assumption or whether it is too strict. Given the assumptions and provided knowledge from System Identification, the essence of LDS is contained in the following two equations:

$$x(t+1) = Ax(t) + v(t), \qquad v(t) \sim \mathcal{N}(0, Q), \tag{4.1}$$

$$y(t) = Cx(t) + w(t), \qquad w(t) \sim \mathcal{N}(0, R). \tag{4.2}$$

In the above equations, $y(t) \in \mathbb{R}^m$ represents an observable state at time t, while $x(t) \in \mathbb{R}^n$ represents a latent state. In the context of videos, $y(t)$ is the vectorization of frame t, i.e. m is denoted as the total number of pixels in a frame. Given this definition for the observable state and latent state, the matrix $A \in \mathbb{R}^{n\times n}$ represents a transition matrix that maps a latent state at time t to a new state in time $t+1$. The matrix $C \in \mathbb{R}^{m\times n}$ then represents an observation matrix - a matrix which denotes the relation between the observable and latent states.

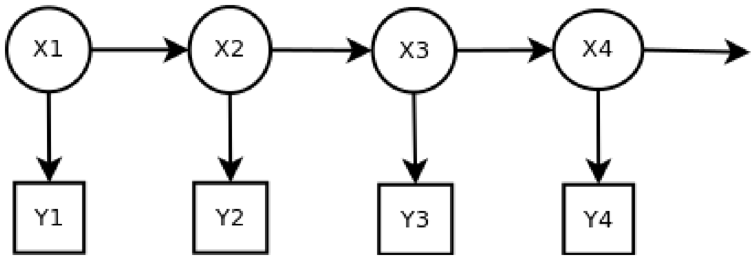


Figure 4.1: Graphical model of a Linear Dynamical System.

Upon observation of the two equations, it can be noted that the LDS model is highly inter-related to other models. For example, the equations also represent a multi-variate version of a first order AutoRegressive Moving-Average (ARMA) model, where 4.1 denotes the moving-average part, and 4.2 denotes the autoregressive part. Furthermore, the graphical model of LDS, shown in Figure 4.1, is similar to a Hidden Markov Model.

The problem of System Identification consists of learning the matrices A, C, Q, and R, as well as the latent states, from only a sequence of video frames. Before discussing the parameter learning stage, it is prudent to first examine the two equations in order to understand what is desired within LDS. In effect, Eq. 4.1 is rather straightforward, since the equation states that a vector at time t is transformed by matrix A into a vector of the same length at time $t+1$, with the addition of white noise to add an element of randomness. Conceptually more interesting is Eq. 4.2. That equation states that a vector y of length m can be represented by a vector x of length n using a transformation matrix C. The main question then becomes what the primary function of the C matrix is. For example, if $m = n$, the C has no function (i.e. becomes the identity matrix), which means that LDS boils down to the first equation.

The primary function of the matrix C is that it serves as a filter for dimensionality reduction. Therefore, C represents a decomposition of the visible state (the video frame) into a set of n filters:

$$y(t) = \sum_{i=1}^{n} \theta_i x_i(t) = Cx(t). \tag{4.3}$$

The dimensionality reduction step is highly desirable, since the visible states are typically very high-dimensional. A video frame of size 640×480 would create a visible state of size 307.200, which in turn results in an A matrix with 94 billion entries, if no dimensionality reduction would have been used. In the original works on dynamic texture recognition, the attention of the dimensionality reduction step is restricted to linear filters. In other words, the goal is to choose an appropriate number of filters n, with $m >> n$. In fact, it is even required that $\tau > n$, with τ the total number of frames of the video.

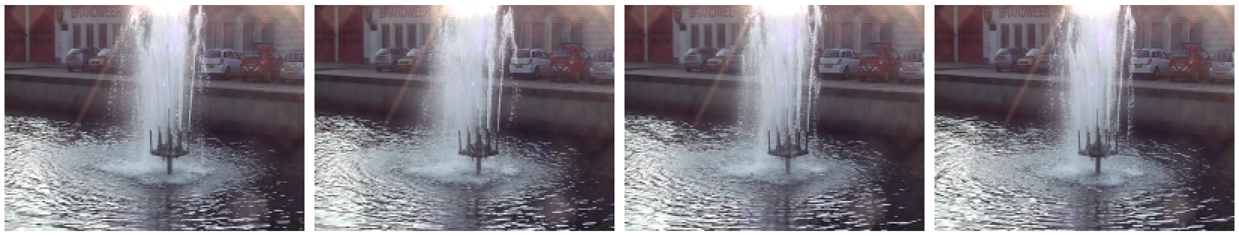
With these assumptions, an intuitive interpretation of LDS for videos can be given. As can be noted from Eq. 4.3, a video frame is represented by a set of vectors $\{\theta_i\}_{i=1}^{n}, \theta_i \in \mathbb{R}^m$, where each filter is weighted by the corresponding element in the vector x. Based on this observation, it is clear that the matrix C consists of n vectors, where the set of vectors can be seen as an eigenbasis of the video. In other words, at each time step t, a video frame $y(t)$ is viewed as a linear combination of the column vectors of the matrix C. Since each column vector is of length m, the vectors can also be viewed as "eigenimages" of the video. This process is exemplified in Figure 4.2. In Figure 4.2a, 4 example frames are shown of a video, while Figure 4.2b shows the "eigenimages" estimated from the whole video (with 200 frames) for $n = 3$. Note that each eigenimage is a column vector of the C matrix. The approach to estimating model parameters such as C is discussed below. Finally, Figure 4.2c shows a typical video frame of the video, along with its approximation and the weight of each eigenimage. Since only 3 eigenimages are used, a level of detail is clearly missing in the approximation.

In the field of System Identification, algorithms to solve the parameter learning problem have been introduced, such as the subspace identification algorithm N4SID [42]. However, these algorithms do not scale well to larger problems such as video modeling. Therefore, suboptimal solutions have been proposed to solve the modeling problem in closed-form [15, 48]. The closed-form solution can be found by first writing Eq. 4.1 and 4.2 in matrix form. Let $Y_1^\tau = [y(1), .., y(\tau)] \in \mathbb{R}^{m \times \tau}$ denote the stacking of the video frames as column vectors into a matrix. This can be done in similar fashion for the latent states, $X_1^\tau = [x(1), .., x(\tau)] \in \mathbb{R}^{n \times \tau}$, and the noise states. The resulting equation in matrix form then becomes:

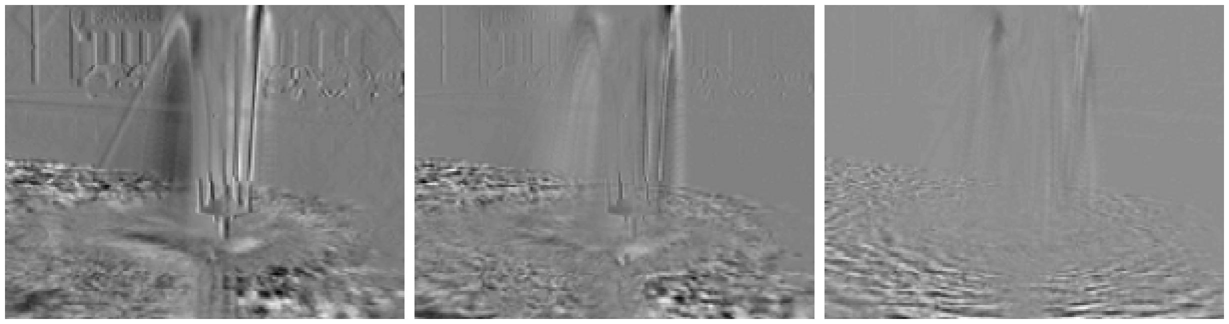
$$Y_1^\tau = CX_1^\tau + W_1^\tau, \tag{4.4}$$

with C in the same form as Eq 4.2. Given that W_1^τ is white noise, the equation in effect states that CX_1^τ is a linear decomposition of the matrix Y_1^τ, i.e.:

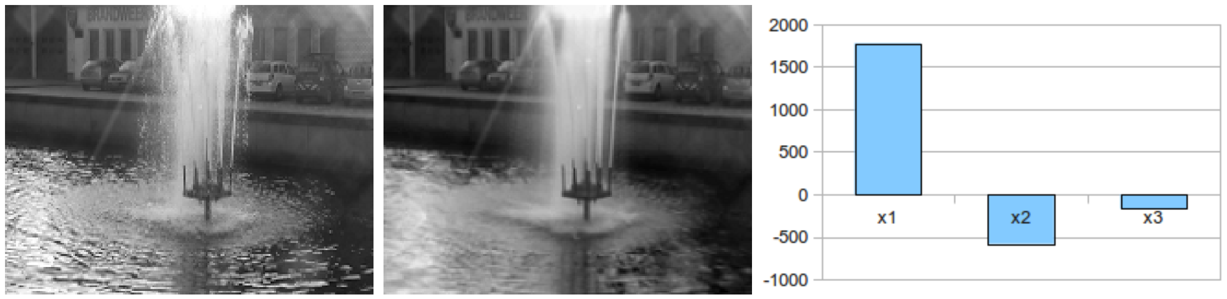
$$C^*, X^* = \underset{C,X}{\arg\min}\, ||Y - CX||. \tag{4.5}$$



(a) 4 example frames of a video. (The colour is provided here for visualization purposes; the input for LDS was a grayscale version of the video.)



(b) Eigenimages for $n = 3$.



(c) Resp. example video frame, low rank approximation, and weight of each eigenimage in the low rank approximation.

Figure 4.2: Visual interpretation of LDS as a linear combination of eigenimages.

The choice for linear filters in the dimensionality reduction ensures that finding a solution for Eq. 4.5 can be done in closed-form. From the theory of Singular Value Decompositions (SVD), it is namely known that the optimal values of a lower rank approximation of a matrix (i.e. a lower rank approximation which is closest to the original matrix) can be found by performing Singular Value Decomposition of the original matrix [21]. Let $Y_1^\tau = U\Sigma V^T$ denote the SVD of matrix Y_1^τ, then $C^* = U$ and $X^* = \Sigma V^T$.

With the values of the latent states known, the first equation of the LDS model can also be solved. Similarly, the equation is first written in matrix form:

$$X_2^\tau = AX_1^{\tau-1} + V_1^{\tau-1}. \tag{4.6}$$

Since $V_1^{\tau-1}$ is again a nuisance parameter and since the values for the X matrices are known, the optimal value for A then becomes:

$$A^* = \arg\min_A ||X_2^\tau - AX_1^{\tau-1}||. \tag{4.7}$$

This equation can be solved by means of ordinary least squares:

$$A^* = (X_1^{\tau-1})^+ X_2^\tau, \tag{4.8}$$

where $(X_1^{\tau-1})^+$ denotes the Moore-Penrose pseudo-inverse of matrix $X_1^{\tau-1}$.

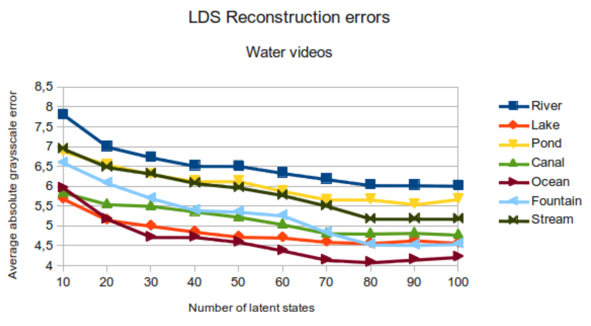
It is furthermore possible to estimate the values of the covariance matrices of the white noise parts of LDS, but this is not done here, since these values hold no interest for discrimination purposes. Due to the use of the SVD, the Linear Dynamical System of a video can also be shortly summarized as Principal Component Analysis with Vector Auto-Regression, as is also noted e.g. in Liu et al. [34].

4.2 Applications to synthesis and classification

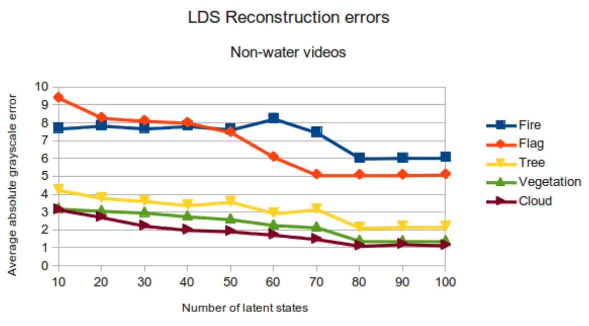
In the above described derivation, the model parameters have been estimated in closed-form, meaning that the "essence" of a video (i.e. a dynamic texture) is captured in the matrices A, C, and X. With these learned models it is possible to serve multiple applications. In this section, two applications are highlighted, namely the creation of infinitely long sequences of a learned model and the discrimination of models for classification purposes. The first application is highlighted since it can provide insight as to what level a LDS model captures the specifics of a dynamic texture. The second application is highlighted since it serves a direct purpose for the water detection problem.

4.2.1 Reconstructing video frames

In general, the ability to reconstruct and synthesize videos serves a wide range of applications, e.g. for simulation or rendering purposes. In the context of this work however, the ability



(a) Water videos.



(b) Non-water videos.

Figure 4.3: The reconstruction error of LDS for the subcategories of the novel database.

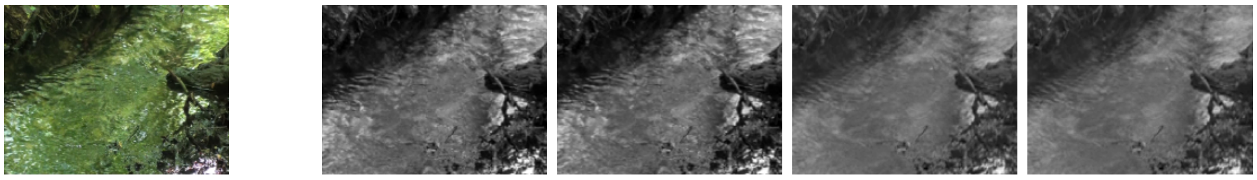
to reconstruct video sequences provides insight into the quality of LDS for video modeling. As can be deduced from Eq. 4.1 and 4.2 and the graphical model from Figure 4.1, all that is required for the synthesis of a video are learned A and C matrices and an initial latent state vector x_0. The quality of reconstruction can then be assessed for a specific video by comparing the original videos and reconstructed videos pixel-wise. In Figure 4.3, the average absolute grayscale error per pixel as a function of the number of latent states is computed for the subcategories of the novel database.

It is hardly surprising that the reconstruction error decreases as the number of latent states increases. From Figure 4.3, it can also be noted that the reconstruction error stabilizes from $n = 80$, while the error is also larger for more chaotic textures such as flags or fire. In Figure 4.4, three example videos are modeled and synthesized using LDS. The Figure contains a frame of the original videos, as well as four synthesis examples, where the examples are taken at four timesteps from left to right. Note how the results are in all cases rather blurry, a process which increases over time. Furthermore, the complex motion of the water ripples and especially the waving flag are troublesome for LDS. From Figure 4.3 and Figure 4.4, it can therefore gingerly be stated that LDS can approximate the spatial and temporal behaviour of dynamic textures, but the approximation comes at the cost of loss in detail and an inability to handle complex and non-linear movement.

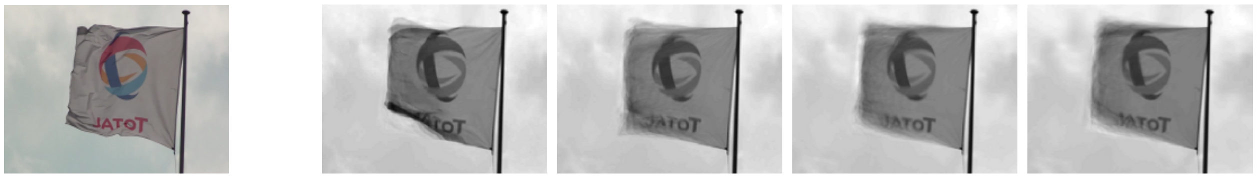
4.2.2 Discriminating LDS models

Although it is obvious that LDS does not perfectly capture textures shown in video material, it is fair to state that that is not required for classification purposes. For classifying textures based on their LDS models, the ability to have small intra-class distances and large inter-class distances is sufficient. Defining a distance measure between LDS models is however non-trivial.

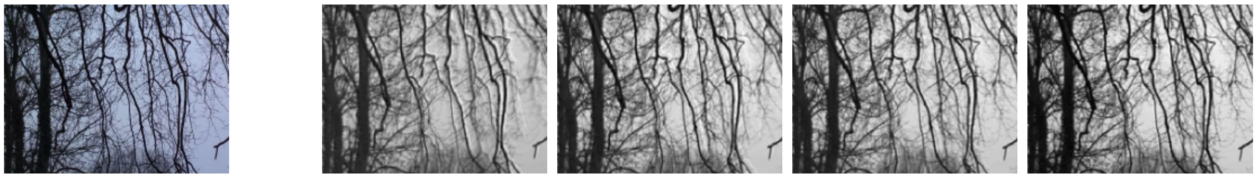
Most works on dynamic texture recognition using LDS advocate the use of the Martin distance for discriminating different models [9, 15, 16, 48]. The Martin distance is a complicated and non-Euclidean distance measure which computes a discrepancy between two



(a) Stream.



(b) Flag.



(c) Tree.

Figure 4.4: Synthesis examples of three videos using LDS for $n = 50$.

LDS models based on the A and C matrices. A primary reason for using such a measure is because the matrix C has a non-trivial geometric structure [15]. The essence behind the Martin distance is based on subspace angles between linear stochastic models.

There are two reasons why this distance measure is often used. First, it is desirable to have a distance measure that exploits both the spatial and temporal mappings (in the form of the A and C matrices). Second, the experimental results on dynamic texture databases containing a single texture per video, such as the UCLA Dynamic Texture Database [48], have been encouraging. However, as will become clear in the next section and in the experimental evaluation, the Martin distance suffers from multiple practical limitations, and therefore it is not used in this work. For a detailed discussion on the use of subspace angles and the discrepancy of learned models, the reader is referred to the work of Doretto et al. [16] and the work of de Cock and de Moor [13].

4.3 System drawbacks and proposed improvements

The focus so far has been aimed at discussing the details of Linear Dynamical Systems, in order to find out whether LDS can provide another perspective to the water detection problem. Based on this discussion and based on experimentation performed with LDS on the water database, multiple fundamental limitations have come to light. In this section, an overview is provided of the main drawbacks of LDS in its original form [15, 48]. Based on these limitations, multiple direct improvements of LDS are provided.

In itemized format, the most fundamental drawbacks of LDS are as follows:

- LDS is performed on whole videos, which means that it can only handle one texture per video. This limitation has severe practical limitations, since natural scenes are complex combinations of textures and objects. Furthermore, experimentation performed on the water database using LDS and the Martin distance indicates that the measure does not yield the desired recognition rates.

- The modeling of whole videos is computationally problematic on the high-dimensional video scale, since e.g. for a video consisting of 500 frames and a window size of 800×600 pixels, the closed-form solutions still requires an SVD procedure on a 480.000×500 matrix, which exceeds the memory of a single machine. Note for example that the videos reconstructed in Figure 4.2 and Figure 4.4 are not only downsized to fit the pages of this thesis, but were also downsized in order to generate a LDS model in a reasonable amount of time.

- The distance measure is complicated and non-Euclidean. More pressing, the advocated Martin distance is very high dimensional and can only compare LDS models of similar frame size. This make the distance computation impossible for videos of different size and is also non-generalizable to non-linear latent variable models such as GPDM [60, 61], that do not rely on linear transformations using the A and C matrices.

- So far, most works on LDS for dynamic texture recognition have focused on either dividing a video into different regions (without stating the target class of each region), or classifying whole videos. The joint problem of division and classification (i.e. classifying each pixel) has not been tackled.

An important notion for these limitations is that they are mostly based on work that is a decade old. In related literature, LDS has since been used for a wider range of problems and it is therefore required to examine to what extend the above listed drawbacks have already been solved. The problem of handling multiple textures within a scene using LDS was already tackled in direct follow-up by Doretto et al. [16]. In that work, LDS was computed for local patches rather than whole videos, and an error functional was used to divide the video into disjoint regions. Note that Doretto et al. were not interested in recovering the target class of each disjoint region, i.e. they did not employ a joint division and classification algorithm as is done in this work. Later work by Chan and Vasconcelos provided a similar approach using Mixtures of Dynamic Textures [9]. On the other hand, in recent work by Chan et al., a hierarchical EM algorithm was introduced to cluster dynamic textures and learning novel textures [7]. They results show that the EM algorithm can detect the presence of multiple textures in a scene, but is incapable to pointing out the locations of the different textures. LDS has furthermore been used to solve other, partially related problems, including background subtraction and saliency detection [37], clustering videos [9], and template tracking [10]. These related works do however not tackle the problems of model learning on high-dimensional videos, the limitations of the advocated distance measure, and the problem of jointly dividing and classifying video frames. Therefore, an attempt has been made in this work to tackle these problems directly.

4.3.1 Scaling LDS to higher quality videos

As stated above, a LDS model can be computed for videos with a low frame size. However, as time passes by, the quality of video sequences increases, which becomes rapidly problematic for LDS. As a running example, consider a video of size 800×600 with 500 frames. In the original LDS formulation, the video is transformed into a matrix $Y \in \mathbb{R}^{(800\times600)\times500} = \mathbb{R}^{480.000\times500}$. From this matrix, the SVD is computed, i.e.:

$$Y = U\Sigma V^T, \tag{4.9}$$

with $U \in \mathbb{R}^{m\times m}, \Sigma \in \mathbb{R}^{m\times t}$, and $V^T \in \mathbb{R}^{t\times t}$. Since LDS is only interested in the first n eigenvectors and -values, nearly all the vectors of the three matrices are clipped. E.g. if $n = 50$, the whole matrix U is first computed, after which all but the first 50 column vectors are removed. Although this is theoretically not a problem, this approach raises serious practical problems. In the running example $m = 480.000$, this means that the U matrix contains over 230 billion entries; a number that easily exceeds the memory of a single machine. As a result, LDS cannot be computed for the video on a single household machine.

The main problem with this approach is clearly that computing the LDS of the original Y matrix is not memory efficient. There is however an observation that can be made regarding the Y matrix, namely that the maximum number singular values of Y is t, since $m >> t$. Therefore, the chosen number of n cannot be greater than t. This observation is an obvious one to make, but can greatly help with solving the memory problem. A well known solution to solve the SVD memory problem for a matrix such as Y is the covariance trick, e.g. performed in the seminal work of Turk and Pentland on Eigenfaces [57]. With the covariance trick, a smaller matrix is computed from the original Y matrix, the SVD is computed from that smaller matrix, and based on that SVD, the SVD matrices of the original matrix can be derived. The full derivation is as follows:

$$\begin{aligned} Y^TY = &\ (U\Sigma V^T)^T(U\Sigma V^T), && Y^TY \in \mathbb{R}^{t\times t}, \\ = &\ (V\Sigma U^T)(U\Sigma V^T), && \\ = &\ V\Sigma U^T U\Sigma V^T, && U^TU = I, \\ = &\ V\Sigma^2 V^T. && \end{aligned} \tag{4.10}$$

In the above derivation, the SVD is computed for the matrix Y^TY, which is much smaller than the original matrix (960 times smaller in the running example). If the final step of Eq. 4.10 is compared to Eq. 4.9, it can be noted that V^T has not changed, while Σ can be computed by taking the square root of the Σ of Y^TY. The only matrix that is missing is U, which can be obtained, since the other matrices are now known:

$$\begin{aligned} U\Sigma V^T = &\ Y && \\ U\Sigma = &\ YV, && V^TV = I, \\ U = &\ YV\Sigma^{-1}. && \end{aligned} \tag{4.11}$$

With the trick derived above, the desired matrices of Eq. 4.9 can be obtained without creating bigger matrices first. Given the SVD matrices, the parameters of the LDS model

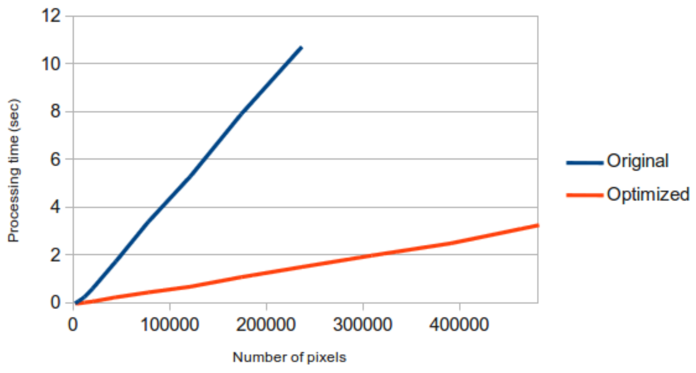


Figure 4.5: Influence of the LDS optimization on processing time.

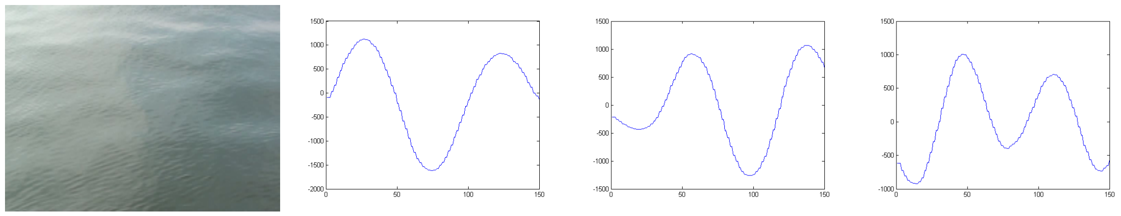
can be computed as explained in Section 4.1. In order to state the practical implications of the optimization step, the optimization is compared to the original algorithm for a single video at multiple scales in Figure 4.5. In the Figure, it is shown that both algorithms have a linear relationship between frame size and processing time. The optimization is however far superior in terms of required processing time. Also note that the original algorithm is limited in the maximum allowed frame size on a single machine.

4.3.2 A novel distance measure between LDS models

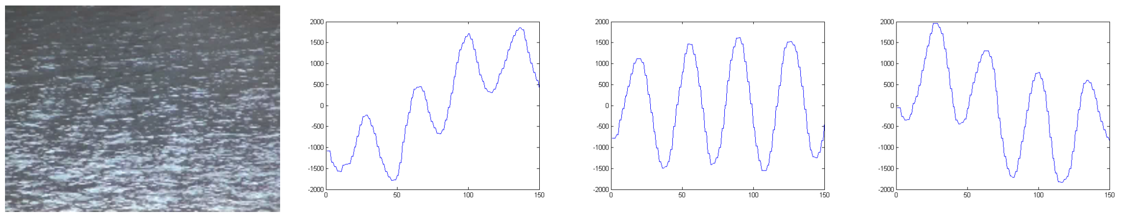
With the optimization in the parameter learning algorithm presented above, models can be learned for large video files. It does however not solve all the memory issues of LDS. A second memory issue arises when attempting to use LDS in a classification environment using the advocated Martin distance. When learning a single LDS model for a whole video, each test video can be classified as being either water or non-water in the context of this work. In order to classify a video, the LDS parameters need to be compared to a collection of training parameters, which means that the A and C matrices need to be known for the train videos.

In other words, the A and C matrices form the descriptor for each learned LDS model. Using the running example of the previous section and using 50 latent states (i.e. $n = 50$), this means that each video has a descriptor of size $\mathbb{R}^{|C|+|A|} = \mathbb{R}^{(480.000\times50)+(50\times50)} = \mathbb{R}^{24.002.500}$. This is of such high dimensionality that classification on a single machine can only be performed for a limited number of videos. Furthermore, if LDS is computed on local patches, e.g. 25×25 patches, the resulting descriptor is still $\mathbb{R}^{(25\times25\times50)+(50\times50)} = \mathbb{R}^{33.750}$. This dimensionality makes it impossible to use LDS in a manner as is done for example with the temporal and spatial descriptors of the previous Chapter.

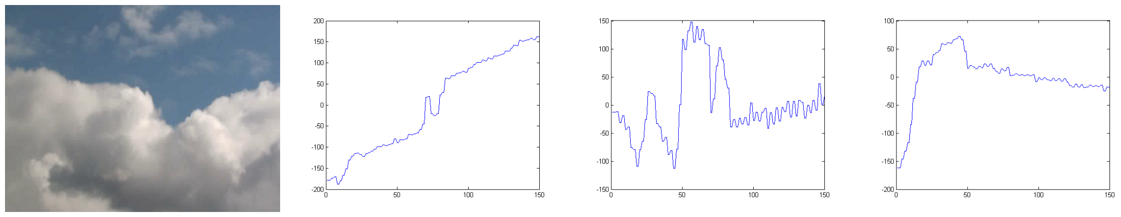
Besides the dimensionality problems of the LDS descriptor when employing the Martin distance, it can also be noted that the distance measure is under-performing compared to other well-known (dynamic) texture algorithms, as well as the algorithm presented in the



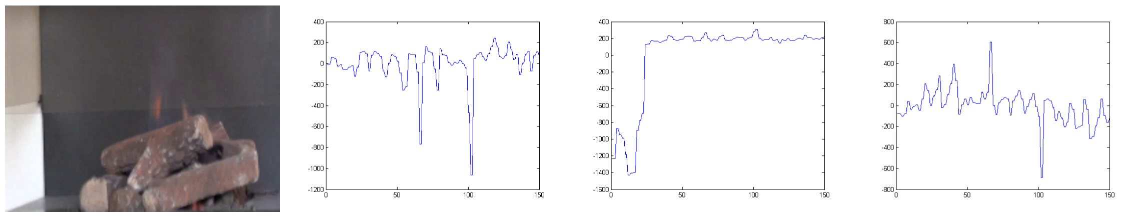
(a) Ocean.



(b) River.



(c) Cloud.



(d) Fire.

Figure 4.6: The first 3 component signals for 4 example videos.

previous Chapter. This point is further discussed in Chapter 5. As a result, if LDS is to be used on a larger scale, a novel descriptor is required. Ideally, the descriptor is naturally of low cardinality and with high discriminative prowess. In addition, a descriptor that does not depend on the A and/or C matrices, but only on the latent mappings X, is generalizable to other latent variable models such as GPDM [60, 61] or GPLVM [26].

Upon investigation of the latent mappings, patterns have been observed that have also been found in the investigation on the local temporal descriptor. Even though the latent mappings are represented by a matrix X in the formulation of Section 4.1, it is also possible to examine the temporal behaviour of individual components. These component signals, i.e. the row vectors of X, state the weight of an eigenimage over time. In Figure 4.6, the component signals of 4 videos are shown for the first three eigenimages.

Similar to the local temporal descriptor of Chapter 3, the component signals are regular, smooth, and periodic. Note however that the signals obtained by the latent mappings of LDS are based on spatio-temporal analysis, rather than purely temporal analysis. A natural idea is to extend the ideas of the temporal descriptor to the component signals by taking the normalized Fourier Transform of each individual signal. In practice, the normalization step has a negative influence on the discrimination quality. For that reason, a descriptor is obtained here from the component signals by taking the regular Fourier Transform for each signal independently. Given a matrix $X \in \mathbb{R}^{n \times \tau}$ computed from a (part of a) video, the Fourier Transform is computed for each of the n row vectors, resulting in a set of Fourier descriptors $\{F^i\}_{i=1}^n$, with $F^i \in \mathbb{R}^\tau$. The final descriptor, dubbed the X-FFT descriptor for the remainder of this work for convenience, is then obtained by concatenating the Fourier descriptors into a single feature vector. Similar to the local descriptors of Chapter 3, two descriptors are compared using the L_2-distance.

The notion of the Fourier Transform is directly derived from the observations in Figure 4.6 and the local temporal descriptor, but it is unknown whether the concatenation of the Fourier Transformed component signals holds any empirical validity. It is for example possible to compare the X-FFT descriptor to the Martin distance by computing the classification rates or Isomap projections on the videos in the water database. A large part of these videos however contain multiple textures and/or objects within a single video, which makes classification and dimensionality on whole videos useless. On the other hand, from existing literature on LDS, it is unknown which local scale is appropriate for LDS. For these reasons, the X-FFT descriptor is validated by examining the segmentation and classification quality using the Pyramid LDS formulation discussed in the next Section.

A second point of discussion is the number of latent components to use. Intuitively, the larger the number of latent components, the higher the discriminative prowess, since that yields additional information. On the other hand, a high value for n results in a high-dimensional descriptor, which has a negative influence on the time required for training. Since the singular values in the SVD of the observation matrix are sorted by importance, it is desired to add the first n components to the X-FFT descriptor. In practice, n is set to 25, since the descriptor becomes saturated for higher values of n.

4.3.3 Pyramid LDS for joint segmentation and classification

With the Introduction of the X-FFT descriptor for LDS models, the segmentation of videos can theoretically be computed by using locally extracted X-FFT descriptors as the local descriptors and by employing the probabilistic classification and spatio-temporal regularization of the algorithm of the previous Chapter. From a practical standpoint, this approach raises serious issues. Most notably, the LDS approach - even with the optimization step and new descriptor - is far too expensive in terms of memory and computation time. This point is further discussed in Appendix A on the complexity analysis of the X-FFT descriptor. Furthermore, in related work on LDS for dynamic texture recognition and segmentation, little attention has been given to the study of the appropriate scale to use LDS. This means that it is unknown whether using LDS solely on local patches, e.g. done in [9, 16], is even appropriate. For segmentation purposes, the scale is ideally as small as possible, in order to ensure a level of detail. It is however possible that LDS cannot capture the specifics of a texture when applied on a local scale.

In light of these issues, an attempt has been made here to jointly investigate the importance of scale in LDS and to create a water segmentation algorithm based on LDS using the X-FFT descriptor. For this investigation, the relaxation has been made that the target class (i.e. water or non-water) does not change over time. This relaxation, which is also applied in multiple related works on dynamic texture recognition [16, 46], ensures that LDS does not need to be computed for a whole set of frames for each video. In order to achieve both goals, the ideas from Lazebnik et al.'s work on Spatial Pyramid Matching [27] are transferred to LDS.

The primary objective of using a spatial pyramid is to examine whether LDS works optimal at a single local scale (like done in Chapter 3), or whether any form of global information holds relevance. More specifically, it is hypothesized here that a combination of local and global information has a positive influence on the segmentation quality of LDS. This is because local scales are more likely to contain a single texture, while global scales can better capture the specifics of a texture.

In the Pyramid Linear Dynamical System (P-LDS) formulation, a video is decomposed into a set of increasingly finer scales. In total, 5 scales are used, as indicated in Figure 4.7. For the 5 scales, the video is decomposed into respectively 1, 4, 25, 100, and 200 parts.

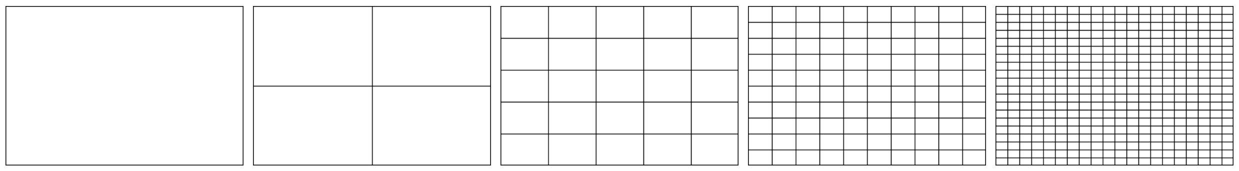


Figure 4.7: The scales used in P-LDS, resp. 0, 1, 4, 9, and 19.

Equipped with the set of scales, the P-LDS algorithm works as follows. For each scale separately, the train videos are first split according to Figure 4.7. Note that the scales divide the video plane only spatially, which means that each part of a video is actually a spatio-temporal cube, defined by the width and height of the part and the number of frames. Throughout the algorithm, the first 200 frames are used. Given the local cubes for the defined scale, the X-FFT descriptor is extracted for each cube, after which the descriptors are fed to a Decision Forest for probabilistic classification. Using a trained Forest, the local cubes at the same scale are extracted for the test videos and then classified. This results in a single heatmap for each test video, where the level of detail is defined by the scale. The heatmaps at the 5 defined scales are shown for a single example of a river in Figure 4.8, where the colour value of each pixel dictates the probability of being water (blue is water, red is non-water). In the Figure, it is shown that finer scales result in a higher level of detail. On the other hand, when LDS is only computed on a fine scale, more obvious global patterns can be overlooked, resulting in outliers. In Figure 4.8, outliers are visible at the finest scale,

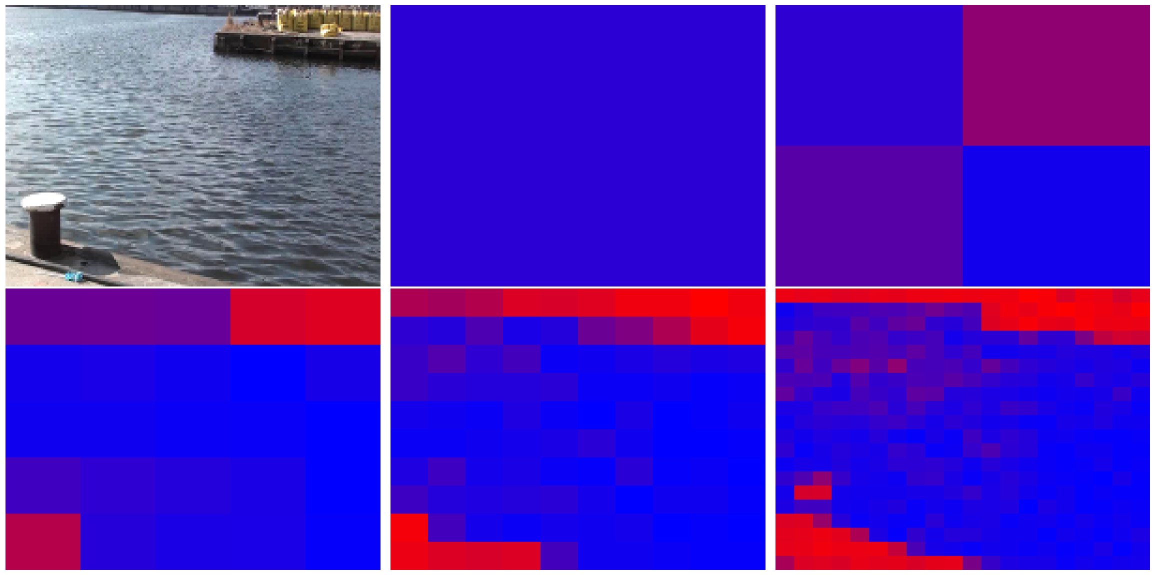


Figure 4.8: P-LDS example.

halfway between the top-left corner and the centre of the of the heatmap.
As a result, the segmentation and classification quality can be determined for each test video at each scale separately, in order to find out which scale is most appropriate. To address the main hypothesis of P-LDS, namely the importance of combining global and local information, the heatmaps can also be combined by created a (weighted) average of the heatmaps. Here, two combinations are explored; the average of the heatmaps and the weighted average based on the scale factor. Given s scales, the average of the heatmaps $\{h_i\}_{i=1}^{s}$ is computed for each pixel (x, y) as:

$$\overline{h_1}(x,y) = \frac{1}{s}\sum_{i=1}^{s} h_i(x,y). \tag{4.12}$$

To compute the weighted average, the weight of each scale is determined based on the scale factor $f_i \in \{0, 1, 4, 9, 19\}$:

$$\overline{h_2}(x,y) = \sum_{i=1}^{s} \frac{(f_i+1)}{\sum_{j=1}^{s}(f_j+1)} h_i(x,y). \tag{4.13}$$

The use of Pyramid LDS serves multiple causes. First, it provides the main goal of this work, namely the detection of water. Second, as stated above, it addresses the questions of scale in LDS. Third, it provides a way to combine the ideas of the local descriptors with LDS. The combination of the local descriptor approach and LDS, further discussed in the experimental evaluation, can be accomplished straightforwardly by combining the heatmap(s) computed for P-LDS and the heatmap for each frame for the local descriptors.

Chapter 5

Experimental evaluation

In this Chapter, the effectiveness of both the local discriminative approach and the generative approach are experimentally evaluated for two tasks on two databases. The first task entails the segmentation of video frames based on the presence or absence of water. This task can also be explained as classifying each individual pixel with respect to water. The second task involves the classification of a whole video (with a binary mask to provide foreground and background pixels) as water or non-water. These tasks are thoroughly evaluated on the water database. In order to further test the quality of the proposed algorithms, the tasks are also performed on a set of selected videos of the DynTex database [44], the current standard for works on dynamic textures.

In the implementation of the segmentation and classification tasks on the databases, the videos of the water database are randomly split with a 60/40 ratio into a train and test set. For each test video, the segmentation is computed for 250 frames. The segmentation fit of the video is then defined as the average of the fit of the individual segmentations with the supplemented binary mask. Given a set of segmentations and a binary mask, the whole video can also be classified as water if the ratio of water pixels in the mask region is at least a half, and non-water otherwise. In order to quantify the variance of the database with respect to the algorithm, the algorithm is run on multiple randomly generated splits and the segmentation and classification results are averaged over the runs.

The rest of this Chapter is outlined as follows. First, the experimentation on video frame segmentation using the water database is discussed, followed by the experimentation on global video classification. After that, the results on the DynTex database are discussed.

5.1 Video frame segmentation

The problem of video segmentation with respect to water presence is discussed here from the two perspectives separately. First, the ability to detect water surfaces in videos is evaluated for the local temporal, spatial, and hybrid descriptors. As stated above, this ability is quantified by comparing computed segmentations to manually created segmentation masks.

More formally, the segmentation fit S of a segmented video V compared to a mask m is computed as:

$$S(V, m) = \frac{\sum_{i=1}^{|V|} s(V_i, m)}{|V|} \times 100\%, \tag{5.1}$$

where $|V|$ denotes the number of frames in the video, V_i denotes the i^{th} frame in the video, and $s(V_i, m)$ is defined as:

$$s(V_i, m) = 1 - \frac{\sum_{x=1}^{W} \sum_{y=1}^{H} |V_i(x, y) - m(x, y)|}{W \times H}, \tag{5.2}$$

with W and H respectively the width and height of the video and the pixel values of the segmentations and the mask are 1 for water and 0 for non-water. In other words, the segmentation fit of a single frame is computed as the complement of the overlay of the frame with the mask, while the segmentation fit of a whole video is computed as the average over the number of frames.

5.1.1 Local descriptor algorithm

In Table 5.1, a numerical overview is provided of the averaged segmentation fit for the individual and combined descriptors. From the Table, it is evident that both the spatial and temporal descriptors are able to robustly segment videos based on the possibility of water. The feature vector combination of the descriptors into a hybrid descriptor has a strictly positive influence on the segmentation quality, indicating that the descriptors are complementary to each other. Furthermore, the use of binary Markov Random Fields for regularization results in a higher segmentation fit for all the descriptors.

Descriptor	No regularization (t=200)	ST-MRF ($\lambda = 1.0, t = 200$)
Hybrid descriptor	90.38% ± 0.5%	**93.19% ± 0.2%**
LBP histograms	85.95% ± 2.4%	90.16% ± 2.3%
Temporal descriptor	83.42% ± 1.3%	87.42% ± 0.8%

Table 5.1: Overview of the segmentation quality of the descriptors.

In Table 5.1, it is shown that 90.38% of the locally extracted hybrid descriptors in the test videos are on average correctly classified (note that the segmentation fit without regularization is similar to the classification rate of the individual descriptors). This is an improvement of 8.34% and 5.15% over resp. the individual temporal and individual spatial descriptors. By employing the spatio-temporal regularization, the segmentation fit can be further increased. Optimal results are achieved when using the hybrid descriptor and the regularization step, yielding an average segmentation fit of 93.19%.

The value λ for the nodes in the MRF model is an influential parameter in the algorithm. In Figure 5.1 the influence of the regularization step with respect to handling outliers and details is shown. If the algorithm is highly dependent on the individual classifications, local

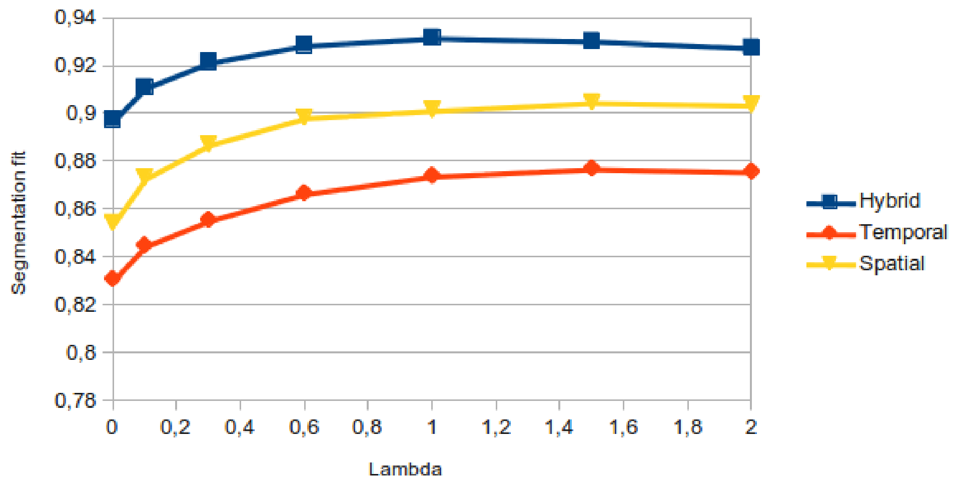


Figure 5.1: Segmentation fit as a function of λ.

parts of the frame can be incorrectly detected as water or non-water. On the other hand, a high dependence of spatial coherence leads to loss of smaller objects and details. Although the optimal value for λ is unique for each video, on average a value of 1.0 meets the trade-off roughly in the middle. Therefore, the reported results and visualizations are yielded for that value.
To further indicate the effectiveness of the discriminative algorithm, example segmentations are shown for multiple videos in Figure 5.2. In Figure 5.2, example segmentations are shown for a wide variety of videos, indicating the ability of the algorithm to detect water in both natural and man-made real-world scenes. Note that the segmentation results do not only show that water textures can be discriminated from non-water textures/objects, both between videos and within the same video, but that it also pushes the limits of ambiguity. This is for example visible in the bottom-right example of Figure 5.2, where the algorithm detects a stroke of water at the location of melting wax.

5.1.2 P-LDS and its combined averages

The approach to evaluating the segmentation abilities is done in similar fashion for the introduced Pyramid formulation of LDS. Given the high computational cost of LDS, the algorithm is run on a single split of the database. In the P-LDS formulation, the segmentation quality is computed for the individual scales, their average, and their weighted average based on their scale factor. In Figure 5.3, an overview is provided of the segmentation quality for P-LDS. The Figure clearly indicates that for the individual scales (blue bars), a finer scale yields a higher segmentation fit with a 13.07% increase between scale 0 (76.39% fit) and scale 19 (86.37% fit). This result confirms the initial belief that the use of LDS at a local scale is beneficial, since local patches/cubes are more likely to contain a single texture.

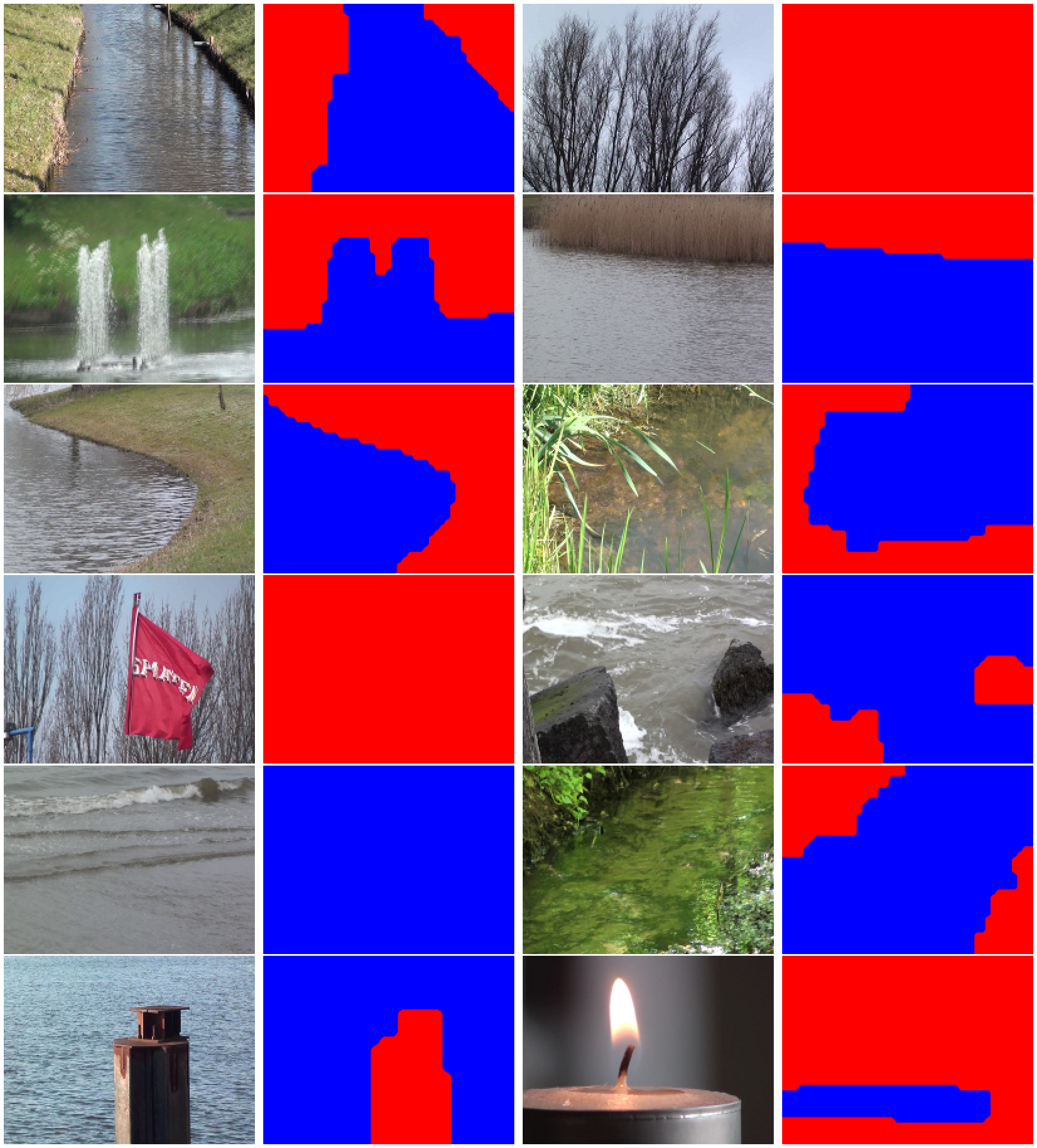


Figure 5.2: Segmentations yielded by the hybrid descriptor for the water database.

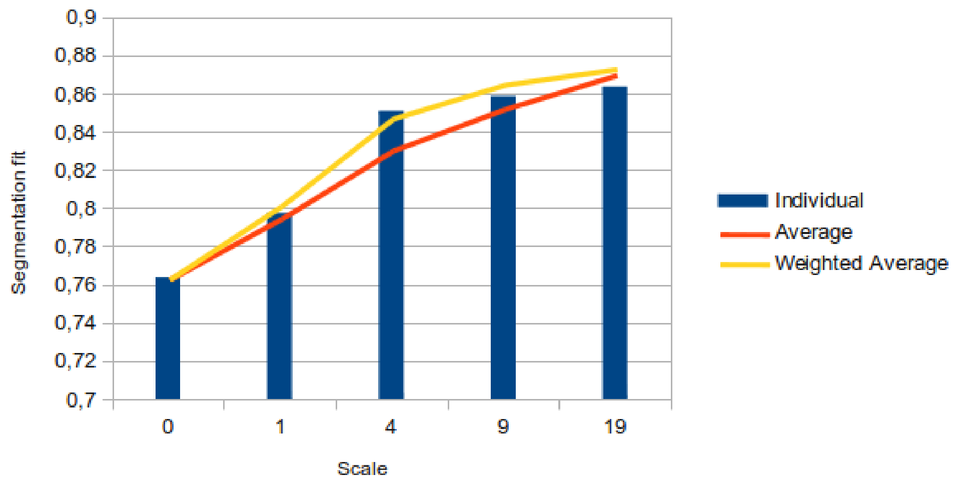


Figure 5.3: Segmentation fit for the individual scales (blue bars) and incremental combinations for the average and weighted average.

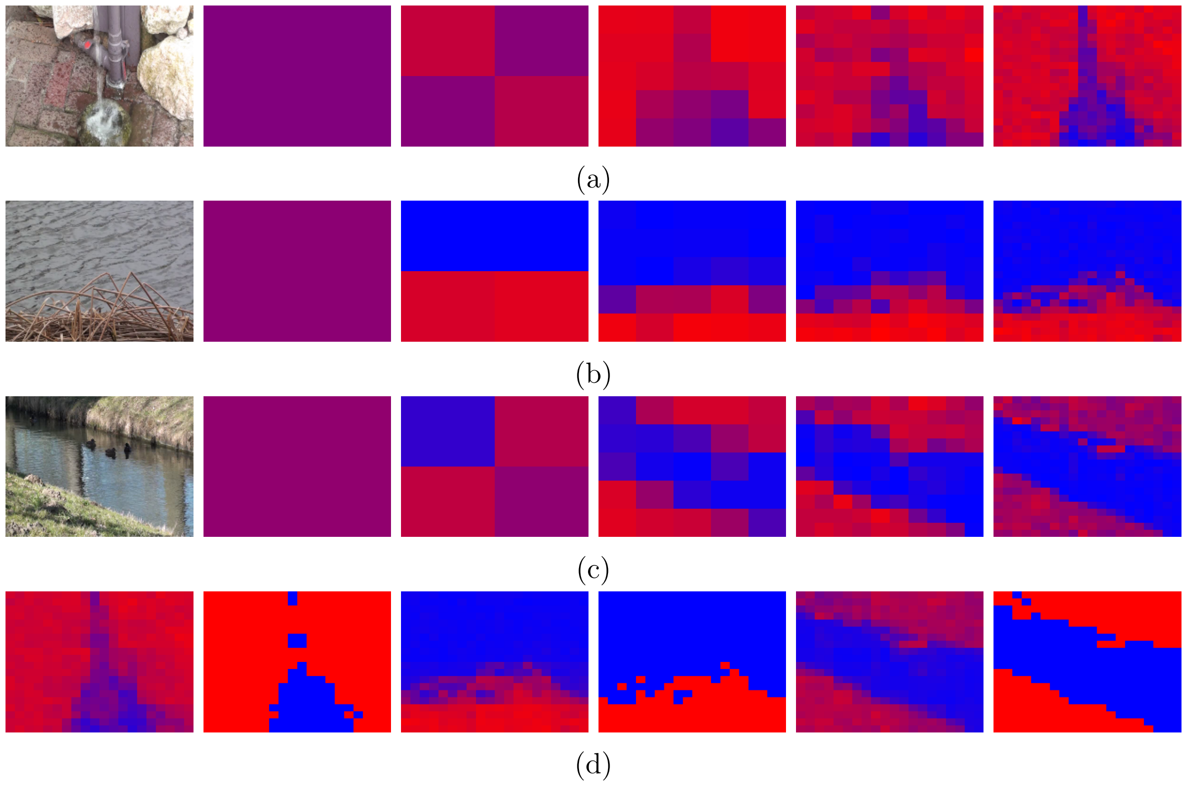


Figure 5.4: Multiple examples of the heatmaps of P-LDS at the 5 scales (a,b,c), as well as the final weighted average and segmentation (d).

In order to evaluate the hypothesis that the combination of local and global information is beneficial, the (weighted) average of the heatmaps are also computed and assessed based on their segmentation quality. In Figure 5.3, the average is shown as the yellow line, while the weighted average is shown as the red line. Contrary to the hypothesis, the global information does not have a strictly positive influence on the segmentation quality.

Note that the lines representing the average and weighted average at each scale denote the (weighted) average of that scale and all the previous (i.e. coarser) scales. However, when all scales are incorporated, both the average and weighted average yield a better segmentation fit than the highest scoring single scale; 86.37% for scale 19, 87.08% for the average of all scales, and **87.38%** for the weighted average of all scales. In Figure 5.4, 3 more examples from the water database are shown, along with the heatmap at each of the 5 scales, the final weighted average, and the binary segmentation based on the weighted average.

From the numerical results, multiple conclusions can be stated regarding P-LDS. In itself, the algorithm has gone beyond other works on LDS in terms of investigating the influence of scale and in terms of creating an algorithm for joint segmentation and classification (i.e. detection). In related works, LDS has so far only been used either for global classification or for dividing the image plane into coherent regions without stating the target class of each region. Note that the work of Ravichandran et al. did provide such an approach [46], but LDS only played a marginal role in their work.

On the other hand, the segmentation results of P-LDS are outperformed by the result of the local descriptors in multiple aspects for water detection specifically. The local descriptors do not only yield higher segmentation fits (93.19% vs. 87.38%), P-LDS is also practically more limited. The practical limits are a direct result of the high computational requirements of LDS. As a result, more time is required to compute a segmentation using P-LDS than with the local descriptors. In addition, in order to diminish the computational cost, the relaxing constraint has been placed on P-LDS that a target class (i.e. water or non-water) does not change over time for each pixel in a video. This relaxation has not been placed on the local descriptors, meaning that the local descriptors have a higher degree of applicability.

5.1.3 Combining both approaches

With the segmentation results of the hybrid descriptor and P-LDS, an interesting question to pose is whether these two perspectives are complementary to each other. This question can be answered by combining the heatmaps for each frame of a video from the hybrid descriptor with the heatmap of the weighted average of P-LDS. The set of combined heatmaps are then served as input to the binary Markov Random Field for regularization. In the objective function of the spatio-temporal MRF discussed in Section 3.5, the prior term does not change. The data term however becomes the weighted average of the probabilities from the hybrid descriptor and P-LDS.

$$V_p(x_p) = \begin{cases} \alpha(1-M_p) + (1-\alpha)(1-H_p) & \text{if } x_p \text{ is water} \\ \alpha M_p + (1-\alpha)H_p & \text{otherwise,} \end{cases} \tag{5.3}$$

where $0 \leq \alpha \leq 1$ denotes the relative weight, M_p again denotes the probability of node p as provided by the hybrid descriptor, and H_p denotes the probability of node p based on P-LDS. Since with the hybrid descriptor, a node p is represented by a pixel value and a frame number, the same node denotes only the pixel values in P-LDS (since P-LDS only provides a single heatmap per video in stead of per frame).

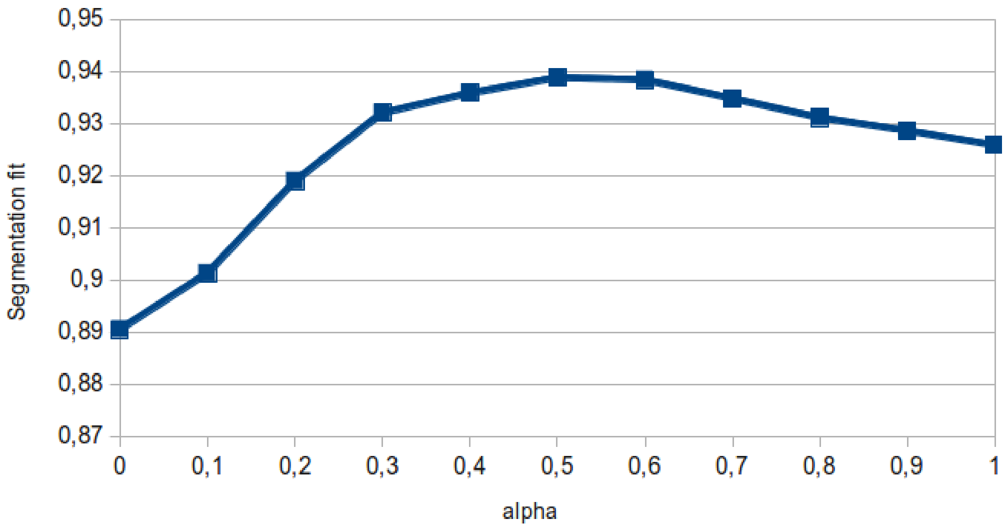


Figure 5.5: Segmentation fit as a function of α.

Given that the segmentation results of the hybrid descriptor are better than the results of P-LDS, it is unknown whether computing the pure average (i.e. $\alpha = \frac{1}{2}$) between both perspectives is ideal. In Figure 5.5, the segmentation fit as a function of α is reported. As can be seen in the Figure, the two methods are indeed complementary to each other, albeit marginally. A final best segmentation fit of **93.89%** is achieved for the combination of both approaches with an α of $\frac{1}{2}$, which is an improvement of 1.4% over the hybrid descriptor on the same split (92.59% segmentation fit). Also note that the result for $\alpha = 0$ is different from the original results reported in the previous section on P-LDS. This is primarily due to the additional use of the spatio-temporal MRF. Although averaging finer scales with coarses scales is a form of regularization, it is still beneficial to explicitly enforce a level of spatial coherence. The segmentation fit of the weighted average of P-LDS with the additional use of Markov Random Fields is 89.05%.

5.2 Global video classification

Although the problem of global video classification is less informative than the segmentation problem, it is still an interesting element of experimentation for two reasons. First, it serves its own set of applications, such as human-aided water detection (i.e. water detection where the user specifies an interesting region). Second, it opens up the possibility for comparison against works from fields such as material and dynamic texture recognition.

As stated earlier in this Chapter, the binary classification of a whole video can be computed by examining the ratio of water pixels in a foreground region. If that ratio is at least $\frac{1}{2}$, the video is classified as water, and it is classified as non-water otherwise. This approach has been opted for both the local descriptors and P-LDS. In order to emphasize the effectiveness of both perspectives in the context of water classification in video, the approaches are compared to several baseline algorithms from the related fields of static material classification and dynamic texture recognition. In total, an in-house implementation has been created for 6 algorithms.

From material recognition, the algorithm proposed by Varma and Zisserman is used [58], which computes a set of filter responses for each pixel as the feature. The filter responses from the train frames are clustered using k-means into 75 clusters, after which the train and test frames are represented by the distribution of the responses with respect to the computed clusters. For the filter bank, the advocated MR8 filter bank is used [1], while the distributions are classified using nearest neighbours and the χ^2 distance.

The other five algorithms are taken from the field of dynamic texture recognition. These algorithms include the original global formulation of LDS using the Martin distance [15, 48], as discussed in Chapter 4, as well as two temporal extensions of LBP, namely VLBP and LBP-TOP [63, 64, 65]. Furthermore, dense optical flow is computed for the videos, after which a signature is created for each video based on 4 flow statistics [43]. The statistics are the average of the curl (1) and divergence (2) of the flow field, and the probability of the field of having a characteristic direction (3) and magnitude (4). The statistics are computed for both the Horn-Schunck flow algorithm [23] and the Lucas-Kanade flow algorithm [36]. For all 5 algorithms, the nearest neighbour approach was used for classification.

An overview of the yielded global binary classification results is provided in Table 5.2. Similar to the results on video segmentation, the temporal and spatial descriptors yield high recognition rates, while their combination yields the best performance, with an average classification rate of **96.47%**. Contrary to Figure 5.5, the classification rate does not increase when averaging P-LDS with the hybrid descriptor. This is primarily because the late fusion of both perspectives only has a modest positive influence on the segmentation fit and therefore also has a minor impact on the classification result.

Comparing the results of both perspectives to the algorithms from material and dynamic texture recognition, it is clear that the hybrid and spatial descriptor outperform all other methods, while the temporal descriptor and P-LDS are only matched by VLBP and LBP-TOP. The experimental results of VLBP and LBP-TOP are hardly surprising, since the pseudo-orderless nature of LBP and its extensions have a longer history of empirical success in dynamic texture recognition in general and have also been successful throughout this work on water detection, as indicated by the results of the spatial descriptor.

Another observation is that the classification rates of the bottom three algorithms, namely LDS with the Martin distance and the flow statistics with both Horn-Schunck and Lucas-Kanade, do not reflect the recognition rates reported in their respective works. For LDS,

[1]The filter bank is available at: *http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html.*

Classification method	Recognition rate
Hybrid descriptor	**96.47% ± 1.6%**
Spatial descriptor	94.23% ± 1.4%
Temporal descriptor	91.88% ± 0.8%
P-LDS weighted average	92.31%
P-LDS average	92.31%
P-LDS scale 19	92.31%
P-LDS scale 9	90.38%
P-LDS scale 4	90.38%
P-LDS scale 1	90.38%
P-LDS scale 0	78.85%
Hybrid descriptor and P-LDS w.avg.	96.15%
Volume LBP [65]	93.51% ± 1.2%
LBP-TOP [64]	93.27% ± 0.8%
MR8 Filter Bank distributions [58]	80.77% ± 6.7%
Linear Dynamical Systems [15, 48]	71.15% ± 2.8%
Horn-Schunck, 4 flow statistics [43]	65.71% ± 2.0%
Lucas-Kanade, 4 flow statistics [43]	61.22% ± 3.4%

Table 5.2: Overview of the recognition rates for binary video classification.

this is mainly because the information of the foreground mask cannot be exploited, since the Martin distance can only compare two LDS models with a similar frame size. In the case of the flow statistics - and also in the case of VLBP and LBP-TOP - there is another possible explanation for the discrepancy between the results in Table 5.2 and the results in their own work. In multiple works on optical flow-based and LBP-based classification, the experiments are performed using a nearest neighbour classification and a leave-one-out test, rather than a single train/test split of the database [19, 20, 43, 65]. In itself, a leave-one-out test is a common approach to classification when dealing with a small database. However, in these works, each video in the used database is first split along the $x-$, $y-$, and $t-$ axis into multiple parts. In other words, from each video, a number of smaller videos are created. Each smaller video is then treated as a separate video, which means that in a leave-one-out approach, each example is not only compared to all the other examples of the database, but also to other parts of the same larger video. This approach creates a tremendous bias in the classification results, since it is likely that different parts of the same video are similar in terms of extracted features. For that reason, the leave-on-out with video splitting approach is not opted here, which generates only modest classification results.

On a more general final note, it is fair to state that the results from the segmentation fits and especially the classification rates are rather high, with a classification rate of over 90% for multiple algorithm. A primary reason for this is that the problem in this work formulated is as a purely binary problem. This means that if for example a tree is classified as fire, it is counted as a correct classification, since the only focus here is the line between water and non-water. The reason that the classification rates of the local descriptors and P-LDS are higher than the segmentation fits is because a perfect segmentation fit is not required to make a correct global classification.

5.3 Evaluation on the DynTex database

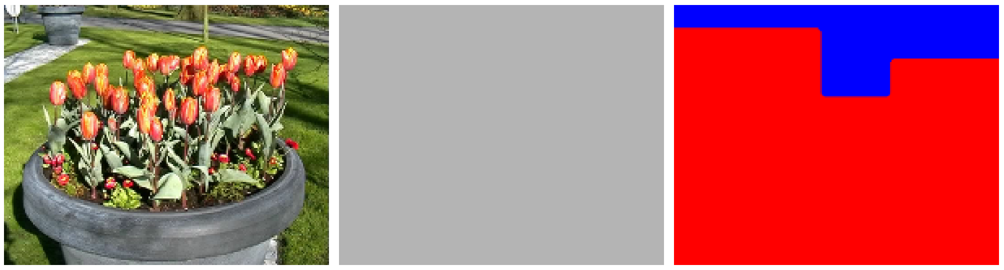
All the experimental evaluation to this point has been performed in the novel water database. In order to further test the quality of the local descriptor and P-LDS algorithms, the segmentation quality and classification rates are also computed on a subset of the DynTex database [44], de facto standard in dynamic texture recognition. Since only a part of the DynTex database contains water surfaces, a subset of 80 water and non-water videos have been selected for evaluation. A second motive for experimenting on the DynTex database is that it provides a comparison for water detection against other non-water textures and objects, such as humans, animals, traffic, windmills, flowers, and cloths. For the classification process, the 80 videos are split into a train- and testset of 40 videos, while the trainset is complemented with an additional 100 videos from the water database.

Used method	Segmentation fit	Classification rate
Hybrid descriptor	**92.68%**	**100%***
Temporal descriptor	84.97%	87.5%
Spatial descriptor	81.29%	87.5%
P-LDS weighted average	80.12%	92.5%
Hybrid descriptor and P-LDS ($\alpha = \frac{1}{2}$)	92.26%	100%*
Volume LBP	-	90.0%
LBP-TOP	-	87.5%
Linear Dynamical Systems	-	75.0%
MR8 Filter Bank distributions	-	72.5%
Horn-Schunck, 4 flow statistics	-	57.5%
Lucas-Kanade, 4 flow statistics	-	55.0%

Table 5.3: Results on the DynTex subset.

The segmentation fits and classification rates, shown in Table 5.3, paint a very similar picture to the results from the water database in terms order of performance. An interesting difference is however that the results are somewhat lower in most cases. This is because the testset of the DynTex selection contains multiple textures that are not present in the trainset. Even more interesting is that the hybrid descriptor is remarkably unaffected. Similarly, roughly 93% of all the pixels are correctly classified, while all videos are given the correct binary label. Observation into the failures of the temporal and spatial descriptors have indicated that the individual descriptors fail on different videos in the testset. Since the early fusion into the hybrid descriptor is not affected by the limitations of the individual descriptors, this is another indication that the temporal and spatial descriptors are highly compatible. In addition, these results state that the hybrid descriptor can generalize more easily to a wider range of negative classes, even when examples are not provided in the train stage. Examples of segmentations are shown in Figure 5.7 for the hybrid descriptor.

The main reasons for the high classification rates of the hybrid descriptor and the fusion of the hybrid descriptor and P-LDS (indicated with the stars in Table 5.3) are similar to the reasons stated in Section 5.2. Again, a person classified as a tree is still considered correct, as long as the water vs. non-water line is not crossed. Also, a segmentation fit which is only partially correct might result in a correct global classification. This point is exemplified in Figure 5.6. As a result, it must be stressed here that the global classification rates are not as informative as the segmentation fits.


(a) Flower example.

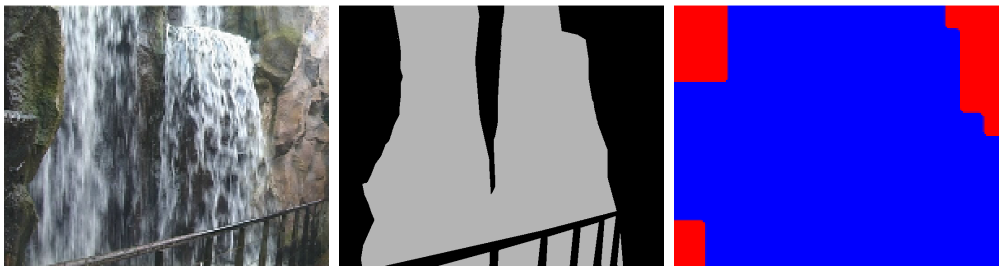

(b) Waterfall example.

Figure 5.6: Two examples of videos with only a partially correct segmentation but a correct global classification. In both cases, the left row shows a sample frame, the middle row shows the manually created segmentation mask (grey=foreground, black=background), and the right row shows a sample segmentation (from the hybrid descriptor). In (a), a non-trivial part of the video is indicated as water (namely the shadows of moving tree branches), but since the ratio is less than a half, the video is classified as non-water. In (b), parts of the background are incorrectly classified as water, but this has no influence on the classification of the foreground pixels. In both situations, local failures are not punished by the classification procedure.

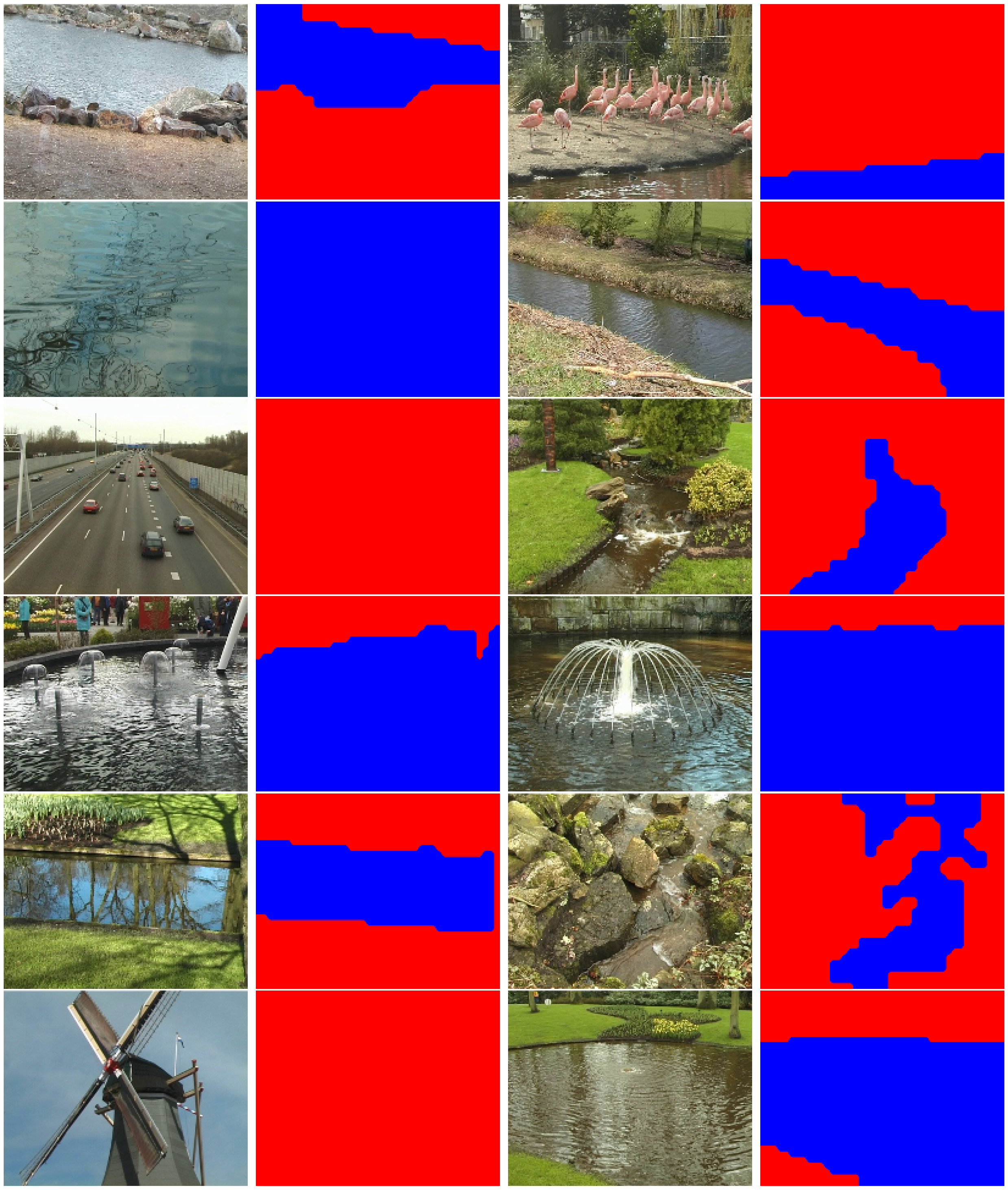


Figure 5.7: Segmentations yielded by the hybrid descriptor for the DynTex database.

Chapter 6

Conclusions

In this work, an attempt has been made to tackle the problem of water detection in videos. Given the lack of attention this specific problem has received, two new algorithms have been proposed for the segmentation and classification of water. The first (discriminative) algorithm introduces a novel descriptor based on the hypothesized temporal behaviour of water and exploits the use of Local Binary Pattern histograms to model the spatial behaviour of water. These descriptors are then directly classified on a local scale and the resulting set of probabilities are regularized to create a coherent segmentation with respect to the presence or absence of water. The individual descriptors and their early fusion into a hybrid descriptor are extracted from pixel-wise mode subtracted colour videos, in order to increase invariance to reflections and inherent water colours.

The second (generative) perspective explores the work on Linear Dynamical Systems for video modeling, an often discussed approach in dynamic texture recognition. Since there is a general lack of analysis and interpretation into the workings of LDS, the elements of LDS have been discussed in this work. Based on this discussion, multiple direct limitations of LDS for water detection have been reported and three improvements have been proposed to subdue these limitations. These improvements are an optimization in the parameter learning stage, a novel descriptor based on the behaviour of the latent mappings of LDS, and a pyramid formulation of LDS (P-LDS). The use of these improvements provides a way to the segmentation of water based on the generative modeling of (parts of) videos.

To test whether the two perspectives are in practice capable of successfully detection water locations, another contribution of this work is the introduction of a new database for the specific problem of water recognition. The database, which will be made publicly available to encourage research into this topic, contains a set of water and non-water videos taken from a wide variety of natural and man-made real-world scenes. Experimental results on this database and a subset of the DynTex database indicate that both perspectives are capable of locally detecting water surfaces in videos. On the water database, an optimal result has been achieved based on the late fusion of the hybrid descriptor and the weighted average of the heatmaps of P-LDS. Since the improvements over the hybrid descriptor are only very modest and given the high computational cost and required relaxations of P-LDS, it can generally be stated the hybrid descriptor algorithm is the most suitable algorithm for water detection.

The reported results bear multiple conclusions. The hypotheses and observations on the temporal behaviour of water, namely a high level of regularity and smoothness, have been empirically validated, given that both the temporal and X-FFT descriptor are directly aimed at capturing these behavioural elements. Furthermore, throughout this work, the temporal and spatial descriptors have shown to be compatible, resulting in a better working hybrid descriptor. As a result, descriptors and a corresponding algorithm have been created which are tailored to the behaviour of water and outperform generic algorithms from related fields. It is hoped that this work has taken a route which leads to further investigations, both to help understand the visual nature of water and to serve multiple direct applications.

6.1 Future work

The algorithms and database presented here reveal multiple aspects for future work, both regarding inherent limitations of the proposed solutions and regarding water detection in general. With respect to the algorithms of this work, multiple elements can be addressed, namely:

- In the current implementation of the algorithms in this work, the descriptors are computed sequentially. A major improvement in speed-up can be achieved by computing the descriptors in parallel. This can greatly decrease the amount of required computational time, which in turn can bring the problem a step closer to (near) real-time detection. Since the regularization step will still be a bottleneck due to the requirement of all probabilities, the combination of a parallel implementation and Geodesic Forests can be rather powerful. The recently introduced Geodesic Forests algorithm of Kontschieder et al. implicitly achieves spatially consistent segmentations by reformulating the training objective of Decision Forests [25].

- In Figures 6.1a and 6.1b, the merits of performing spatio-temporal regularization using ST-MRF is shown. As indicated earlier in this work, the regularization takes care of outliers and miss-classifications, which generates a coherent segmentation. On the other hand, the enforcement of spatial coherence of pair-wise neighbours treats small and/or thin objects as outliers. The drawbacks of this approach are indicated in Figures 6.1c and 6.1d. In order to keep the merits of regularization without the inherent limitations, richer models are required.

- The algorithms based on the local descriptors and P-LDS can discriminate water from other textures and objects, but they have a restricted set of possible applications. This is mostly due to the number of consecutive frames required to create both the temporal descriptor and the X-FFT descriptor of LDS. As a result, a relatively long exposure to a texture or object is required for identification. An obvious improvement is an extension of the algorithm or a novel descriptor that requires only a short exposure.

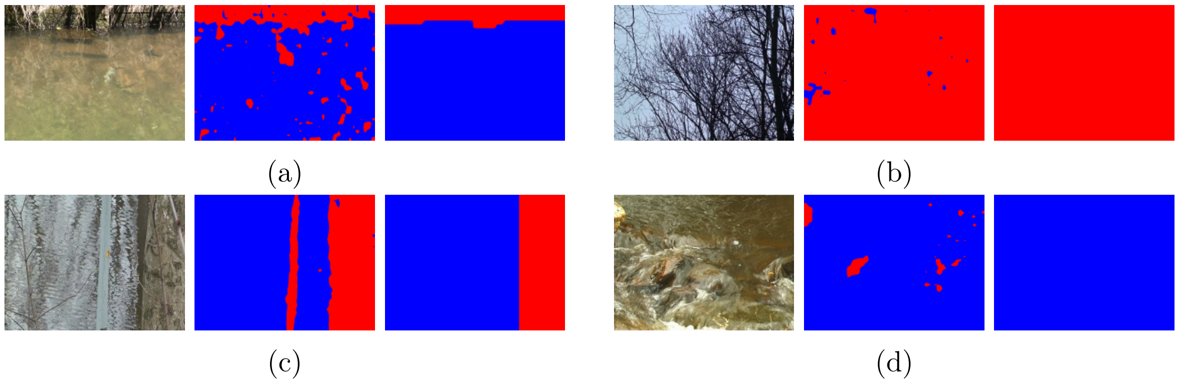


Figure 6.1: A few examples of the merits of regularization (a,b) and the drawbacks (c,d). Each example shows a frame and a segmentation without and with regularization. In (a) and (b), the spatio-temporal MRF cleans up the segmentations by removing unwanted outliers. In (c) and (d) however, thin and small objects which are originally classified as non-water (resp. the street post in (c) and the rocks in (d)) are treated as outliers and also removed.

From a more general scope, there are multiple aspects which can be explored:

- Although the numerical results reported in this work are high, they hardly state that water detection is a solved issue. Not only do the algorithms require a long exposure as stated above, they are also computationally expensive and it is unknown what effect camera motion has on the detection quality. In order to make further progress into water detection, elements such as real-time detection, large-scale recognition (i.e. recognition in very large sets), and detection with a moving camera need to be addressed. It is merely hoped that the algorithms of this work can provide a basis for later work and that the water database can be useful for parameter learning and algorithm testing.

- Another aspect which can be explored is to what extent the proposed LDS improvements in the form of the parameter optimization, the X-FFT descriptor, and Pyramid LDS are useful for dynamic texture recognition in general. Furthermore, throughout this work, it has been hinted that video modeling does not need to be restricted to linear latent variable models. Current literature on dynamic texture recognition based on video modeling are all about improving LDS for better results, but the step to non-linear latent variable models such as GPDM [60, 61] have so far hardly been investigated.

- It is lastly also interesting to note that the videos of the water database can be useful as an extension of dynamic texture databases.

Appendix A

Complexity analysis of the X-FFT descriptor

The high computational complexity of the introduced X-FFT descriptor has been stressed multiple times and it is therefore good practice to examine the time and space complexity of computing a single X-FFT descriptor. Since the descriptor is based on the temporal descriptor of Section 3.2, the complexity of the temporal descriptor is first computed. The temporal descriptor is yielded from a patch of size $m \times m$ for t consecutive frames by computing the mean brightness value for each of the t patches, followed by a Fast Fourier Transform (FFT) and a normalization step. Assuming a naive approach to computing the mean brightness value by going through all m^2 pixels, the complexity of the temporal descriptor is:

$$O(tm^2 + t\log t + t) = O(tm^2 + t\log t), \tag{A.1}$$

given that the FFT of a t-dimensional descriptor can be computed in $O(t\log t)$. Computing the X-FFT descriptor of the same patch size however takes a greater number of steps. Using the optimization of Section 4.3.1, the dot-product between the matrix derived from the spatio-temporal patch and its transpose needs to be computed. From the resulting matrix, the SVD needs to be computed (c.f. Eq. 4.10). After that, the X matrix is computed as the dot-product between the matrices Σ and V^T. The last step entails computing the FFT of the n row vectors of X.

The first step entails computing the dot-product between $Y^T \in \mathbb{R}^{t\times m^2}$ and $Y \in \mathbb{R}^{m^2\times t}$, which can be done in $O(tm^2m^2t) = O(t^2m^4)$ time. From the resulting matrix $Y^TY \in \mathbb{R}^{t\times t}$, the computation of SVD has a time complexity of $O(4t^3 + 8t^3 + 9t^3) = O(t^3)$.

The third step involves a dot-product between $\Sigma \in \mathbb{R}^{n\times n}$ and $V^t \in \mathbb{R}^{n\times t}$, which can be done in $O(n^3t)$ time. Computing the FFT of the n row vectors is then computed in $O(nt\log t)$ time. The final time complexity then becomes:

$$O(t^2m^4 + t^3 + n^3t + nt\log t). \tag{A.2}$$

Since $n \in O(t)$, the equation can be simplified to:

$$O(t^2m^4 + t^3 + t^3t + t^2\log t) = O(t^2m^4 + t^4). \tag{A.3}$$

Furthermore, where the space complexity of the temporal descriptor is $O(t)$, the space complexity of the X-FFT descriptor is $O(nt)$. The time and space complexity of the X-FFT descriptor give a clear indication of why the descriptor cannot simply be plugged into the algorithm presented in Chapter 3 and why additional relaxations are required to make water detection possible.

Bibliography

[1] S. Beauchemin and J. Barron. The computation of optical flow. *ACM Computing Surveys*, 27(3):433–466, 1995.

[2] S. Bochkanov and V. Bystritsky. Alglib (www.alglib.net).

[3] A. Bosch, A. Zisserman, and X. Munoz. Scene classification via plsa. *European Conference on Computer Vision*, pages 517–530, 2006.

[4] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence*, 2004.

[5] Y. Boykov, O. Veksler, and R. Zabih. Markov random fields with efficient approximations. In *Computer vision and Pattern Recognition*, pages 648–655. IEEE, 1998.

[6] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[7] A. Chan, E. Coviello, and G. Lanckriet. Clustering dynamic textures with the hierarchical em algorithm. *Computer Vision and Pattern Recognition*, pages 2022–2029, 2010.

[8] A. Chan and N. Vasconcelos. Mixtures of dynamic textures. *International Conference on Computer Vision*, 1:641–647, 2005.

[9] A. Chan and N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. *Pattern Analysis and Machine Intelligence*, 30(5):909–926, 2008.

[10] R. Chaudhry, G. Hager, and R. Vidal. Dynamic template tracking and recognition. *International Journal of Computer Vision*, pages 1–30, 2012.

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

[12] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends in Computer Graphics and Vision*, 7(2):81–227, 2012.

[13] K. De Cock and B. De Moor. Subspace angles between linear stochastic models. In *Decision and Control*, volume 2, pages 1561–1566. IEEE, 2000.

[14] E. Dinic. Algorithm for solution of a problem of maximum flow in networks with power estimation. In *Soviet Math. Dokl*, volume 11, pages 1277–1280, 1970.

[15] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

[16] G. Doretto, D. Cremers, P. Favaro, and S. Soatto. Dynamic texture segmentation. *International Conference on Computer Vision*, 2:1236–1242, 2003.

[17] S. Dubois, R. Péteri, and M. Ménard. A comparison of wavelet based spatio-temporal decomposition methods for dynamic texture recognition. *Pattern Recognition and Image Analysis*, 5524:314–321, 2009.

[18] S. Fazekas, T. Amiaz, D. Chetverikov, and N. Kiyyati. Dynamic texture detection based on motion analysis. *International Journal of Computer Vision*, 82(1):25–32, 2009.

[19] S. Fazekas and D. Chetverikov. Normal versus complete flow in dynamic texture recognition: a comparative study. *International Workshop on Texture Analysis and Synthesis*, pages 37–42, 2005.

[20] S. Fazekas and D. Chetverikov. Analysis and performance evaluation of optical flow features for dynamic texture recognition. *Signal Processing: Image Communication*, 22:680–691, 2007.

[21] G. Golub and C. Van Loan. Matrix computations, 1989.

[22] Z. Guo, L. Zhang, and D. Zhang. A completed modeling of local binary pattern operator for texture classification. *Image Processing*, 19(6):1657–1663, 2010.

[23] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981.

[24] D. Hu, L. Bo, and X. Ren. Toward robust material recognition for everyday objects. *British Machine Vision Conference*, pages 48.1–48.11, 2011.

[25] P. Kontschieder, P. Kohli, J. Shotton, and A. Criminisi. Geof: Geodesic forests for learning coupled predictors. *Computer Vision and Pattern Recognition*, 2013.

[26] N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *Advances in neural information processing systems*, 16(329-336):3, 2004.

[27] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition*, 2:2169–2178, 2006.

[28] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44, 2001.

[29] L.-J. Li and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. *International Conference on Computer Vision*, pages 1–8, 2007.

[30] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing. Object bank: A high-level image representation for scene classification & semantic feature sparsification. *Advances in neural information processing systems*, pages 1378–1386, 2010.

[31] S. Li. *Markov Random Field Modeling in Computer Vision.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1995.

[32] S. Liao, M. W. Law, and A. C. Chung. Dominant local binary patterns for texture classification. *Image Processing*, 18(5):1107–1118, 2009.

[33] C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz. Exploring features in a bayesian framework for material recognition. *Computer vision and Pattern Recognition*, pages 239–246, 2010.

[34] C.-B. Liu, R.-S. Lin, N. Ahuja, and M.-H. Yang. Dynamic textures synthesis as nonlinear manifold learning and traversing. *British Machine Vision Conference*, pages 859–868, 2006.

[35] L. Ljung. *System identification.* Wiley Online Library, 1999.

[36] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[37] V. Mahadevan and N. Vasconcelos. Spatiotemporal saliency in dynamic scenes. *Pattern Analysis and Machine Intelligence*, 32(1):171–177, 2010.

[38] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[39] A. Mumtaz, E. Coviello, G. Lanckriet, and A. Chan. Clustering dynamic textures with the hierarchical em algorithm for modeling video. *Pattern Analysis and Machine Intelligence*, 35(7):1606–1621, 2013.

[40] R. Nelson and R. Polana. Qualitative recognition of motion using temporal texture. *CVGIP: Image Understanding*, 56(1):78–89, 1992.

[41] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[42] P. Overschee and B. De Moor. N4sid: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30(1):75–93, 1994.

[43] R. Péteri and D. Chetverikov. Dynamic texture recognition using normal flow and texture regularity. *Pattern Recognition and Image Analysis*, 3523:223–230, 2005.

[44] R. Péteri, S. Fazekas, and M. Huiskes. Dyntex: A comprehensive database of dynamic textures. *Pattern Recognition Letters*, 31(12):1627–1632, 2010.

[45] A. Rankin and L. Matthies. Daytime water detection and localization for unmanned ground vehicle autonomous navigation. *Proceeding of the 25th Army Science Conference*, 2006.

[46] A. Ravichandran, P. Favaro, and R. Vidal. A unified approach to segmentation and categorization of dynamic textures. *Asian Conference on Computer Vision*, 6492:425–438, 2011.

[47] J. Ren, X. Jiang, and J. Yuan. Dynamic texture recognition using enhanced lbp features. *International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[48] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. *Computer Vision and Pattern Recognition*, 2:II–58–II–63, 2001.

[49] R. Schwind. Polarization vision in water insects and insects living on a moist substrate. *Journal of Comparative Physiology A*, 169(5):531–540, 1991.

[50] L. Sharan, C. Liu, R. Rosenholtz, and E. Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, pages 1–24, 2013.

[51] L. Sharan, R. Rosenholtz, and E. Adelson. Material perception: What can you see in a brief glance? [abstract]. *Journal of Vision*, 9(8):784, 2009.

[52] A. Smith, M. Teal, and P. Voles. The statistical characterization of the sea for the segmentation of maritime images. In *Video/Image Processing and Multimedia Communications*, volume 2, pages 489–494, 2003.

[53] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. In *ACM international conference on Multimedia*, pages 399–402. ACM, 2005.

[54] L. Spencer, M. Shah, and R. Guha. Determining scale and sea state from water video. *Image Processing*, 15(6):1525–1535, 2006.

[55] J. Tenenbaum, V. de Silva, and J. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[56] J. Tessendorf. Simulating ocean water. *Simulating Nature: Realistic and Interactive Techniques. SIGGRAPH*, 2001.

[57] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[58] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1):61–81, 2005.

[59] M. Varma and A. Zisserman. A statistical approach to material classification using image patch exemplars. *Pattern Analysis and Machine Intelligence*, 31(11):2032–2047, 2009.

[60] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

[61] J. Wang, A. Hertzmann, and D. Blei. Gaussian process dynamical models. In *Advances in neural information processing systems*, pages 1441–1448, 2005.

[62] S. Zhang, H. Yao, and S. Liu. Dynamic background modeling and subtraction using spatio-temporal local binary patterns. In *Image Processing*, pages 1556–1559. IEEE, 2008.

[63] G. Zhao and M. Pietikäinen. Local binary pattern descriptors for dynamic texture recognition. *International Conference on Pattern Recognition*, 2:211–214, 2006.

[64] G. Zhao and M. Pietikäinen. Dynamic texture recognition using local binary patterns with an application to facial expressions. *Pattern Analysis and Machine Intelligence*, 29(6):915–928, 2007.

[65] G. Zhao and M. Pietikäinen. Dynamic texture recognition using volume local binary patterns. In *Dynamical Vision*, pages 165–177. Springer, 2007.