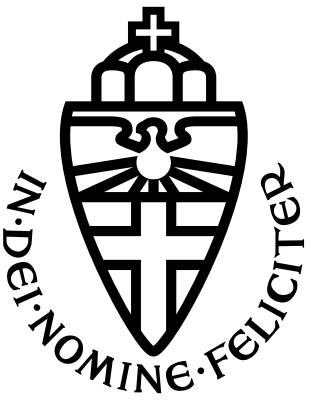
Master Thesis



Radboud University

Image Based Time Synergy

Author:
Merel Witsenboer
s4577027

First Supervisor/Reader:
prof. dr. ir. A.P. de Vries
Second Reader:
prof. dr. M.A. Larson

February 26, 2021

Disclaimer

This thesis is subject to a non-disclosure agreement, meaning that some details and sections have been left out. The full thesis is available upon request.

Abstract

This thesis addresses the problem of *Image Based Time Synergy*, where photos are clustered in groups based on the events during which these photos were taken. Rather than clustering the photos based on their timestamps, they are clustered based on their visual content instead. Neural Networks are used to extract temporal information for each individual image, and to predict the relative temporal order within each pair of images. Then, the problem of image based time synergy for chronologically ordered images is addressed. The boundaries between events are determined based on the visual similarity between pairs of images. At first, photos are clustered on one level, making every event boundary equally important. Next, a hierarchical event cluster tree is created, where boundaries higher up in the tree represent the greatest dissimilarities between images, and are thus the most important event boundaries. Boundaries in the bottom of the tree help identify sub-events. The initial lists of boundaries as well as the final trees are compared to the results of an existing method that creates time-based hierarchical cluster trees using timestamps rather than the visual content. Several experiments are carried out to determine the best settings for the image-based event clustering algorithm.

Contents

Chapter 1

Introduction

Photos are a means to capture and remember important parts of our lives. Rodden [2002] notes that "the most important use of digital photographs is to record holidays or other significant events, and then show the pictures to friends or family". These pictures capture one or more specific events, such as a wedding, a holiday, or a day to the zoo. Some of these events, such as a holiday to France, contain multiple smaller events. Someone could, for example, go to the beach on one day, and visit Paris the day after. People tend to sort their photos in a chronological order, rather than mingling photos of the Eiffel Tower with beach photos. Also, people prefer to group their photos based on events, which is also shown in Rodden and Wood [2003].

But as it happens, people take a lot of pictures, as many as the storage on their phone or camera allows. Manually arranging this enormous set of photos takes up too much time, and so an algorithm was developed to take over. Such an algorithm should work in a similar manner as the users themselves. Whatever algorithm replaces the user, should also be able to group the photos per event and sort them chronologically.

As could be expected, it is much more difficult for a machine to guess which photos belong to which events, or to guess how many different events there are to begin with. Thus far, the machine is capable of determining events, as long as the photos contain timestamps. Although most modern devices keep track of time, the timestamps themselves are not always correct. Modern mobile phones often automatically adjust to the current time and the current time zone, ensuring that the timestamps for your photos are correct. However, not all photos are taken with modern devices, and several major problems can mess up the timestamps for photos. Firstly, for many devices, the default timestamp is the year 1970, based on the Unix time system[1], which is not always updated by the device itself or its user. Secondly, not every device automatically updates the timestamp when travelling across timezones, which becomes a problem when the user wants to merge photos from multiple different devices. Thirdly, iPhones cause a loss of metadata when using photos in non-native applications. Finally, Social Media platforms may cause problems. Many users upload their photos to Social Media platforms such as Facebook, and then delete the original photos from their device. The

[1] Unix time system briefly explained: `https://www.unixtimestamp.com/`

problem here is that the original timestamp could be lost after uploading, depending on the specific Social Media platform.

It is indisputable that a new technique is desired, one that would be able to predict the events to which the photos belong, even in the absence of temporal metadata. This problem will be described as *image based time synergy* in this thesis. Image based time synergy is about using visual content rather than metadata to determine which events took place, and during which event each individual photo was taken. This thesis will take a first step into solving this problem by trying to determine the events in a chronologically ordered set of photos. A more detailed description can be found in Section 2.3. This thesis attempts to answer the following research question:

> **RQ**: *To what extent is it possible for an algorithm to solve the problem of image based time synergy for chronologically ordered images in the absence of further temporal information?*

This research question consists of multiple parts. The main question is if it is possible for an algorithm to solve the problem of image based time synergy. This question is reduced to chronologically ordered photo sequences, as will be explained in Section 2.3. The algorithm developed in this thesis focusses on event clustering in the absence of further temporal information, so the ordering of the images is the only temporal information used by the algorithm. The approach taken to solve this problem is described in Chapter 4.

The hypothesis is that it should be possible for an algorithm to solve this problem, although the performance is expected to be lower than the performance of timestamp-based approaches, since there is less information available to work with. Trying to solve the problem for unordered images, which would be the logical next step, would be even more difficult for the same reason. Within this thesis, the hypothesis will be tested by developing a technique for image based time synergy for chronologically ordered images, where photos are clustered per event, independent of further explicit temporal information. As such, the term *image based time synergy* will be used to describe the problem for chronologically ordered images only.

The work in this thesis is divided in two parts. First, an attempt is made to predict temporal information based on visual content. This problem is divided in three parts. At first, the problem of Season prediction is addressed, where a model is trained to predict the season during which a photo was taken. Next is the problem of Part of Day prediction, where a model is trained to determine the part of the day during which a photo was taken. Finally, the relative order of two images is predicted, also called A before B prediction. The question formulated for this part of the thesis is:

> **Q1**: *To what extent is it possible for an algorithm to predict temporal information in terms of season, part of day and relative order, based on visual content only?*

The hypothesis is that a model should be able to predict both the season and the part of day during which a photo was taken. Predicting the relative order of two images is expected to be more difficult. The approach to solve each of these three problems is explained in Chapter 3.

The second part of this thesis addresses the problem of image based time synergy itself, by clustering the photos per event based on the visual content. The question for this second part is as follows:

> **Q2**: *To what extent can image similarity in combination with background knowledge acquired from many photo sequences help identify which photos belong to which event?*

The background knowledge mentioned in this question ranges from the knowledge about the temporal order of the photo sequences, to the use of the predictions about the temporal information from Chapter 3. This question specificially addresses the use of image similarity, where images are compared based on their visual content. The hypothesis is that computing image similarity is a very suitable approach when using visual content for the task of event clustering. The approach is further discussed in Chapter 4.

Chapter 2

Background

2.1 Related Work

This chapter describes some of the most relevant research done for the clustering of photos for events. Most of this research was done for photo browser systems. Although there is very little research done for event clustering in the absence of temporal information, the papers desribed in this chapter give some interesting insights and ideas.

Jaimes et al. [2000] have presented STELLA, to provide the user with clusters that can be used to create digital albums. This may already be a step in the direction of event clustering, although STELLA does not explicitly try to take temporal information into account. For example, photos of flowers may be grouped into one cluster, simply because they all contain flowers. It does not matter when each photo was taken, they could even be taken years apart. They make use of the concept of Recurrent Visual Semantics (RVS), where certain objects or scenes are present in multiple photos. Bracketing is an example of RVS, where multiple images are exactly the same, although with different exposure settings. Finally, photos are clustered based on content features such as color and composition.

Rodden [2002] have investigated several visual features that may be useful for sorting photos. They note that most photo browsers sort thumbnails – smaller versions of the actual photos – in a default order, and that alternatives have not been investigated much. Similar to textual documents, photos could be sorted based on some similarity measure. For example, visual features such as color, could be used to determine similarity between photographs. Visual content can be described on different levels, namely based on colours and textures, on objects and activities, on places and people, or on the feelings induced by the specific photo. They describe several features for similarity measures, such as the average color, region summaries, and varying color histograms. Again, this approach does not explicitly try to detect events, but their idea to use similarity measures may be useful.

Graham et al. [2002] have taken a different approach, focussing on the time difference between sequential photos. They investigated the issue of current photo browsers,

that have limited tools for organization of photos. Even though this approach may not be useful in this research, given that it requires concrete temporal information, the authors provide some interesting insights for event clustering. They have developed two browsers for photo collections that exploit the time dimension of photos. Their approach to event clustering is based on the assumption that photos from one event are taken very close in time, and further away in time from photos from other events. They describe the high number of images within an event as a burst, and they describe sub-events as sub-bursts. These sub-bursts are also encoded in the cluster structure. A tree of clusters is created, where each burst can consist of multiple sub-bursts. The actual photos are stored at the leaf nodes. Such a tree is the end goal of this research as well, although the concept of bursts cannot be used directly due to the absence of temporal information.

Rodden and Wood [2003] investigated how people use photo browser systems, such as Shoebox. Their research reveals how users prefer to have their photos ordered, and what kind of retrieval tasks they perform. They reported that most albums created by participants are classified by some event. Within such an event-related album, photos are most often sorted chronologically. They also found three main arguments for searches done by the participants. They were either looking for photos taken during a certain event, for a single specific photo, or for a set of photos from different events with for example a specific person present. For each of the three tasks, having the photos grouped per event makes the search easier. This research emphasizes the importance of event clustering in general.

There are several techniques that attempt to create event clusters, however most of them rely on temporal information and possibly other metadata. For example, Cooper et al. [2005] have developed multiple methods to cluster photos based on temporal information and visual content. These methods either rely only on temporal information or on both temporal information and visual content. Using only visual content or visual similarity to discover events is rather difficult, given that photos within an event can vary. For example, photos of the same place can vary depending on the time of day, and a zoo often has both desert and tropical landscapes. The only useful feature then seems to be the corresponding timestamps.

However, even with timestamps, it may be difficult to determine events. Both a day at the zoo and a three week vacation can be seen as one large event, although of different duration. While it makes sense that a three week vacation has multiple sub-events, such as a day to the beach and a day to the mountains, a day to the zoo also has sub-events. At the zoo, one could say that having lunch is an event, and that visiting the zebras is an event as well. Different (sub-)events have different durations, making it more difficult to determine when an event begins and when it ends. Cooper et al. [2005] quantify similarity between timestamps by comparing them in a pairwise fashion. The resulting similarity matrix shows higher similarities for photos that are closer in time. Next, they use a checkerboard pattern to identify the event boundaries, where the center of this pattern is the boundary between photos from two events. To locate these boundaries, they compute a novelty score, where peaks indicate which boundaries are likely. Taking into account that events can have very different durations, they use multiple time scales to compute the similarity.

The approach taken in this thesis is mostly based on the work of Loui et al. [2005]. They use foreground segmentations to find event clusters. For foreground segmentation, they use a block-based technique to eventually obtain regions that are either foreground or background. Once these regions have been established, successive images are compared using a similarity measure. This measure then helps to determine if the images are likely to be taken during the same event or not. The similarity measure is based on features extracted from the regions, such as luminosity, color, position and size. This approach only depends on timestamps to chronologically order the images, but is otherwise independent of temporal information. For image based time synergy for ordered images, this approach can be very useful. For undordered images, it might be necessary to make pairwise comparisons across all images, rather than just comparing successive images. Using foreground segmentations for event clustering is an interesting approach, given that the background within an event often remains the same. Foreground objects, such as people, may also be present across a number of images within an event.

Although Loui et al. [2005] have their own approach to foreground segmentation, other techniques have been developed. Ang Lim and Yalim Keles [2018] have implemented a triplet Convolutional Neural Network (CNN) combined with a transposed CNN to obtain foreground segmentations as well. Convolutional Neural Networks, and Neural Networks in general, will be explained in Section 2.2. More details on the work done by Loui et al. [2005] and Ang Lim and Yalim Keles [2018] can be found in Section 2.3.

For evaluation of the technique developed in this thesis, an existing time-based method is used. This method returns hierarchical event trees. From these trees, event boundaries can be derived. More information on the similarity matrices and hierarchical trees can be found in Section 2.3.2.

2.2 Neural Networks

Neural Networks can learn to recognize patterns in data, and can be used for classification and regression tasks. Neural Networks can be used to learn complex non-linear relationships and often turn out to generalize well to unseen data[1]. The basic architecture is explained in this section, as well as some more specific Neural Networks such as the Convolutional Neural Network and Siamese Neural Network.

A basic neural network consists of inputs, hidden units and output units. An example neural network is given in Figure 2.1. Each connection between two units has a weight, and each unit has a bias. Bishop [2006] provides a detailed overview and explanation of the architecture and equations involved. The hidden units compute activations for the inputs, that are then fed to an activation function. The outputs of this function are the inputs for the next layers. The activations are computed using Equation 2.1.

[1]For some more general information on Neural Networks, visit `https://towardsdatascience.com/introduction-to-artificial-neural-networks-ac338f4154e5`

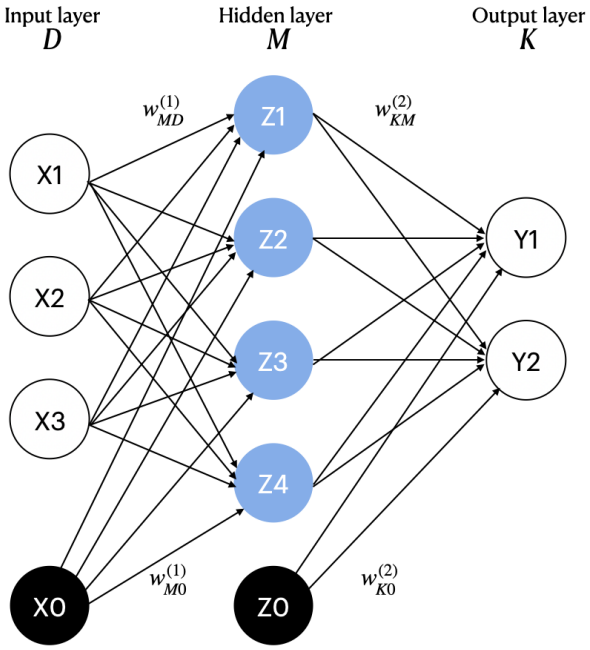


Figure 2.1: Example of a basic Neural Network with one hidden layer. This figure was inspired by Bishop [2006] (p. 228)

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{2.1}$$

The activation outputs are transformed using a nonlinear activation function h, following eqation 2.2.

$$z_j = h(a_j) \tag{2.2}$$

The superscript (1) denotes that the parameters belong to the first layer. $j = 1, ..., M$ are the linear combinations of the input variables. $i = 1, ..., D$ are the input variables x. $w_{ji}^{(1)}$ are weights, while $w_{j0}^{(1)}$ are biases. These activations are computed for every hidden unit using the inputs, biases and the weights. For the output activations, the equation is slightly adapted, as can be seen in Equation 2.3.

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{2.3}$$

where the superscript (2) denotes that the parameters belong to the second layer. $k = 1, ..., K$ are the output units, and z_j is the output of an activation function. The final output of the neural network is often transformed using a sigmoid function for binary classification, or using a softmax function for multiclass classification.

While the process of feeding the inputs through the network towards the output

units is known as forward propagation, backpropagation distributes prediction errors over the responsible units and adjusts weights and biases to minimize this error.

2.2.1 Convolutional Neural Networks

Bishop [2006] also gives an overview of Convolutional Neural Networks (CNNs). A CNN consists of (one or more) sets of Convolutional and Sub-Sampling layers. Each unit in the convolutional layer has a local receptive field of the input, based on the concept of neighbouring pixels having a higher correlation than distant pixels. Each convolutional layer may contain multiple feature maps, since multiple different features should be detected.

Such networks are invariant to several input transformations. All units will treat each part of the input in the exact same way, which provides the basis for the invariance of the network. A Convolutional layer has fewer weights than fully connected layers, and there are restrictions on the updating of these weights, reducing the number of parameters to be trained.

The Sub-Sampling layers of the CNN further reduce the number of parameters. This layer again has a receptive field, and may take the sum, average or maximum value of the input within the receptive field. The receptive fields, in this case, do not overlap. With a 2x2 size, the output will be half the width and height from the input. In practice, the sub-sampling layer is a pooling layer, and is often a Max Pooling layer. The Max Pooling layer takes the maximum value of each receptive field.

Finally, the output of the final Sub-Sampling layer is generally fed to a fully connected layer. This layer is responsible for making the predictions based on the detected features.

2.2.2 ResNet

A network that is used throughout this thesis is the Residual Network (ResNet). ResNets contain Residual Blocks, where identity mapping is the most important part. These Residual Blocks help training very deep neural networks, making them perform better than smaller counterparts. Without Residual Blocks, the performance of deeper networks seems to drop. One major problem is the vanishing gradient, where the gradient of the loss function approaches zero when more layers are added to the model. With identity mapping, the output of a previous layer is added to the output of a layer further down. More details on these ResNets and the Residual Blocks can be found in the works of He et al. [2016a] and He et al. [2016b] respectively.

Several variants of ResNet are available online, such as a model with 50 layers (*ResNet50*), as well as models with for example 18 or 152 layers (*ResNet18* or *ResNet152* respectively). As will be explained in Chapter 3, a *ResNet50* will be used, pre-trained on existing datasets. The concept of pre-training is explained in Section 2.2.5.

2.2.3 MobileNet

For the final sub-task of Chapter 3, a MobileNet is used instead. A MobileNet is generally a smaller network that has been optimized on speed of inference. Rather than combining all three channels (RGB), convolution is done on each channel separately. Pointwise Convolution then combines the outputs of these convolutions. More detailed information about MobileNets and their architecture can be found in the work of Howard et al. [2017]. A MobileNet pre-trained on existing datasets will be used in this research.

2.2.4 Siamese Neural Networks

Siamese Neural Networks consist of two or more identical sub-networks. They provide a solution when there is very little or incomplete data available. First explained by Bromley et al. [1993], this network structure enables the model to determine new classes by computing the similarity between a single data point and the currently existing classes. When this similarity is low, the model is able to create an entirely new class, based on just this one data point. The model does not need to be retrained on all data, which is the main advantage of this structure.

To determine if two images are similar, the data is split in pairs, creating a new label describing the distance between each two images. Some pairs contain images belonging to the same class, while other pairs contain images from different classes. The model is trained to give two outputs, one for each image in a pair. The loss function is then used to determine the distance between the outputs and to compare this distance to the actual distance label. This loss function is also known as the *contrastive loss*.

For this research, a Siamese Neural Network seems suitable and hopefully useful to determine the chronological order between images. The approach is further explained in Section 3.4.

2.2.5 Pre-Training Models

The Neural Networks explained in the previous sections are used in this research, although pre-trained on one or more existing datasets. The first dataset used for pre-training is *ImageNet*, created by Deng et al. [2009]. The other main dataset used to pre-train the models from this research on is *Places365*, the core set of the Places2 dataset created by Zhou et al. [2017]. While ImageNet contains images corresponding to nouns, such as *dog* or *chair*, the Places365 contains images for different scenes, such as bedrooms or streets. The final dataset used by the pre-trained models is *Landmarks*, created by Noh et al. [2017]. This dataset contains images of landmarks around the world, labeled with the names of these landmarks.

Pre-trained models are already trained on a dataset for a different problem. For example, the ImageNet dataset can be used to pre-train a model. The model will have learned important visual features for classifying these images with the corresponding labels (nouns suchs as *dog* or *chair*). While these labels are useless for this thesis, the features are not. Replacing the final prediction layer (where the model decides if the image represents a dog or a chair), with a layer that makes the desired predictions,

will work well assuming that the dataset used for pre-training is similar to the dataset used for the actual problem. The same set of features will be extracted by the model to make predictions for the new dataset. Using a pre-trained model also saves a lot of time and resources, since most of the network parameters do not have to be trained from scratch. The most important layer to be trained is the new prediction layer.

2.3 Image Based Time Synergy

The problem of *image based time synergy* can be split in two main components: for chronologically ordered photos, and for unordered photos. Note that the latter will be the most difficult problem and solving it would imply a solution for the first problem as well. A solution to the first problem will already be highly valuable, considering that the problem of event clustering itself will then have been solved for a common case of ordered photo streams. A solution to the second problem is of course most desirable, considering that uploaded photos cannot be assumed to be chronologically ordered. In this thesis, however, the problem will be addressed for chronologically ordered images only.

This section further analyses the research done for the main components that will be used for image based time synergy. The approach taken by Loui et al. [2005] will be explained in detail. However, since the approach for foreground segmentation is replaced by the work of Ang Lim and Yalim Keles [2018], their work is explained first. Finally, the relevant work for image based time synergy for an unordered set of images is discussed.

2.3.1 Foreground and Background Segmentation

Ang Lim and Yalim Keles [2018] have created *FgSegNet*, a triplet CNN combined with a transposed CNN, to obtain foreground and background segmentations. A triplet CNN can be described as three CNNs, where the outputs of each CNN are combined. These three CNNs are based on a pre-trained VGG-16 Net, although the final parts are slightly adapted. For example, dropout regularization is applied to avoid overfitting. The three CNNs consist of exactly the same blocks and share network weights. A transposed CNN can be described as a CNN that works the other way around: it performs decoding instead of encoding. More details on the triplet CNN and transposed CNN can be read in the work of Ang Lim and Yalim Keles [2018].

Their network is trained on CDNet2014, created by Wang et al. [2014], which containes pixel-based ground truth segmentations. Their method has been evaluated on several challenging categories, such as *bad weather* and *low frame rates*, and has outperformed all existing methods thus far. The dataset consists of frames obtained from video feeds, which is the main difference from actual photographs.

Each photo is fed to the neural network with two duplicates. These duplicates are the exact same image, but on a different scale. The different scales are obtained using a Gaussian pyramid with a sigma based on a downscaling factor. The first part of the network is the triplet CNN, as mentioned earlier. Each of the different scaled photos is fed to one of the three CNNs, resulting in three feature embeddings. The embeddings

are then combined by upscaling the downscaled embeddings to match the scale of the embedding of the original photo. The combined feature map is fed into the final part of the network, which is the transposed CNN. The resulting mask is exactly the size of the original photo and shows the foreground and background segmentiations.

The output masks of the *FgSegNet* are passed on to determine events, explained in the following section.

2.3.2 Event Clustering using Foreground Segmentation

Differences between Foreground and Background Regions

Loui et al. [2005] use multiple features to compute the distance between individual regions in successive images. These features are computed first for foreground, and then for background. The differences between the foreground and background regions are computed to compare successive images.

For each region, the luminosity is computed. The luminosity formula for a specific pixel is given in Equation 2.4. The mean luminosity is taken for each region separately. Different regions can then be compared by taking the difference of the mean of the first region and the mean of the second region.

$$y = 0.299R + 0.587G + 0.114B \tag{2.4}$$

Next, several other features are computed. The computations of the intensity, saturation and hue are given in Equations 2.5, 2.6 and 2.7.

$$Intensity\ (I) = \frac{R+G+B}{3} \tag{2.5}$$

$$Saturation\ (S) = 1 - \frac{R+G+B}{I} \tag{2.6}$$

$$Hue\ (H) = \cos^{-1}\left(\frac{\frac{1}{2}(R-G)+(R-B)}{((R-G)^2+(R-B)(G-B))^{\frac{1}{2}}}\right) \tag{2.7}$$

The hue, intensity and saturation are combined into a color set component for each region. These features are then used to compute the distance between two regions. Loui et al. [2005] have provided the full equations, although some components remain unclear. Every foreground region of one image is compared to every foreground region of another image, and the same goes for the background regions.

Once individual regions from successive images have been compared in a pairwise manner, a total distance between all regions in the images is computed. The harmonic mean, given in Equation 2.8 is used to compute this total distance, where a_i is the distance between the individual regions.

$$harmonic\ mean\ (a_1, a_2, ..., a_n) = \frac{1}{\frac{1}{a_0} + \frac{1}{a_1} + ... + \frac{1}{a_n}} \tag{2.8}$$

Event Clustering

The total distances computed in Equation 2.8 are then used to determine the event clusters. Loui et al. [2005] use a threshold, all distances above this threshold are taken as the event boundaries. Distances below the threshold will not be taken as boundaries, and the corresponding images thus belong to the same event. The threshold is to be determined, and could be a function of the discovered distances (using for example the average values and standard deviation).

To prevent a cluster from being split into two clusters based on a *one-off* image, distances between chronologically more distant photos can also be used. Rather than only taken the adjacent images, the model could also skip one or two images for every comparison.

Precision, recall and the F_1-score are used to evaluate the event clusters. The equations are given in Equations 2.9, 2.10 and 2.11 respectively. The equations were adapted from Cooper et al. [2005].

$$precision = \frac{\#correctly\ detected\ boundaries}{\#total\ number\ of\ detected\ boundaries} \tag{2.9}$$

$$recall = \frac{\#correctly\ detected\ boundaries}{\#total\ number\ of\ ground\ truth\ boundaries} \tag{2.10}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.11}$$

The approach for event clustering will be used for chronologically ordered sets of images. Including the order in which images were taken is an interesting easier step before solving the issue for unordered images. The algorithm will only need to determine where the event boundaries are, and does not need to worry about the timeline of the images. In Chapter 3, an attempt is made to predict some temporal information based on visual content. Then in Chapter 4, the approach for image based time synergy for chronologically ordered images is explained.

Ground Truth Event Clustering

To evaluate the algorithm developed in this thesis both quantitatively and qualitatively, a ground truth is developed using an existing method. Using the chronologically ordered timestamps for each set of photos, hierarchical trees and the corresponding event boundaries are obtained. These trees and boundaries are used as ground truth in the experiments in Chapter 4. An example tree is given in Figure 2.2.

It should be noted that the ground truths used in this thesis are not 100% accurate. The exact performance of the ground truth method on our dataset is unknown, since there are no event labels are available.

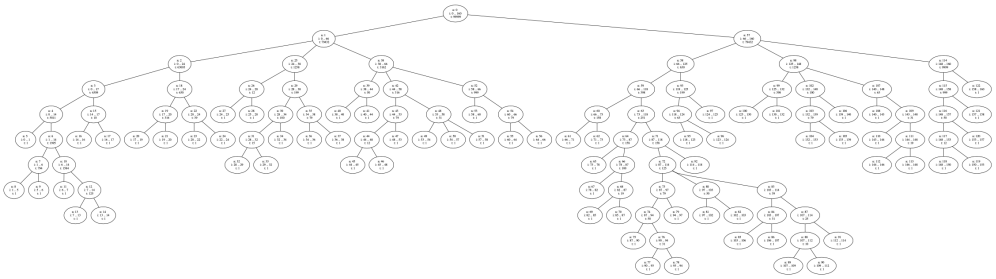


Figure 2.2: Full time-based hierarchical tree for a set of 160 photos.

Chapter 3

Predicting Temporal Information

To get an intuition of the complexity of the final objective, image based time synergy, a number of presumably simpler problems are explored. Based on the visual content, an attempt will be made to predict temporal information in three steps: Season, Part of Day, and A before B, in order of expected difficulty. Assuming that these problems are solvable, the predictions may be useful to help solve the problem of image based time synergy.

While this section explains in detail the data and approaches used to solve the problem of predicting temporal information, Chapter 4 explains the approach for solving image based time synergy for chronologically ordered images.

3.1 Data

An existing dataset is downloaded and used for this research, which contains multiple photos grouped per user. For Season prediction, Part of Day prediction, and A before B prediction, the timestamps are necessary to determine the labels. For Season prediction, the geographic location is required as well, since seasons depend upon the hemisphere where the photo was taken. Hence, the timestamps, longitudes, and latitudes are downloaded along with the photos. The photos are also filtered based on the exact metadata values, and based on the source of the photos. Only uploaded photos are allowed, preventing photos from sources such as Facebook to be downloaded. As discussed earlier, photos from some sources, such as Social Media platforms, may contain incorrect timestamps. Also, only photos taken after 1970 are allowed, to avoid downloading photos that have a default timestamp. Finally, the photos are also filtered on latitude and longitude. Any photo without geographic metadata is given a latitude and longitude of 0.0 by default, so only photos with longitudes and latitudes greater or smaller than 0.0 are allowed. It is worth noting that a lot of data is disregarded using these filters, leaving some collections with as few as just one or two photos. In total, 267296 images are downloaded.

Hemisphere	Months	Season
Northern	December-February	Winter
	March-May	Spring
	June-August	Summer
	September-November	Autumn
Southern	December-February	Summer
	March-May	Autumn
	June-August	Winter
	September-November	Spring

Table 3.1: Season labels based on month and hemisphere

After downloading, the labels are computed for the specific task. More details on these labels can be found in the corresponding sections below. The data is split into training, validation and testing of sizes 70%, 20% and 10% respectively. The split is created in such a way that all photos from the same user stay together within one of these three sets. Requiring photos of individual users to be kept together makes it impossible to have an exact 70/20/10 split, but the effect of this minor deviation is expected to be negligible. After the data has been split, the photos are shuffled, preventing similar photos from staying together during training.

In a final procedure, the dataset is enlarged through data augmentation using image flipping, rotation and zoom operations. Images are flipped horizontally, but not vertically as the model is expected to base its predictions on for example the sky. Images are rotated up to 15 degrees. Other augmentations are a zoom range of 0.1, a shear range of 0.05, a width shift range of 0.05 and a height shift range of 0.05. Finally, the data is balanced to avoid overfitting on a certain class.

Specific data preprocessing steps for each of the three tasks, as well as details on the creation of labels, can be found in the corresponding sections below.

3.2 Season Prediction

3.2.1 Data

For this task, the data needs to be preprocessed even further. Any photo taken somewhere close to the equator cannot be used for Season prediction, since this area does not have seasons. Photos taken in the Arctic or Antarctic Circle are also removed for the same reason. The remaining photos are those taken at a latitude between 23 and 65 degrees, both on the Northern and Southern hemisphere.

Next, the labels for Season prediction are computed. The labels are 0 for *spring*, 1 for *summer*, 2 for *autumn* and 3 for *winter*. The label assignment based on the month and hemisphere can be found in Table 3.1. This assignment is based on the meteorological seasons rather than the astronomical seasons. The balanced data contains, for a batch size of 32, 3657 batches for training, 1115 batches for validation and 584 batches for testing, equally spread across the four seasons.

3.2.2 Training Procedure

For Season prediction, a *ResNet50* is used. This ResNet contains 50 layers, and a top layer is added for classification. This model is pre-trained on ImageNet and expects normalized data. For normalization, the input images are first converted to a range of $[0, 1]$, next they are normalized with the required values[1]. A linear classification output layer is added, with four output units. The cross-entropy loss function is used, combined with an $L1$ regularisation term. This regularisation term is multiplied with 0.00001 and summed with the cross-entropy loss. For optimization, the standard Stochastic Gradient Descent (SGD) is chosen, with a weight decay of 0.00001 for $L2$ regularization. The learning rate is set to 0.01, which is the usual starting point for a learning rate for SGD, and the momentum is set to 0.9. SGD does not compute the exact derivative of the loss function, but estimates the derivative based on a small batch of data, meaning that the derivatives are slightly off. Momentum is used to compute weighed averages, resulting in better estimates of the derivative, leading to faster convergence during training[2].

Finally, a learning rate scheduler is used as well, that reduces the learning rate whenever the validation loss is stuck on a plateau. The learning rate is reduced with a factor 10, with a patience of 5, meaning that the learning rate is not reduced earlier than 5 epochs without improvement. The minimum learning rate is set to 0.00001. During training, the model keeps track of the total number of epochs during which the validation loss did not improve. When this counter reaches 15, the model stops training. The model is trained for at most 100 epochs. Given that the model seems to converge within 25 epochs, this is a safe maximum that will never be reached.

All parameter values chosen in this research are based on commonly used values in similar models. For optimal predictions, these parameters have yet to be tuned. Parameter optimalization is considered out of scope for this thesis.

Next, a second *ResNet50* is trained. This second model is pre-trained on Places365 rather than ImageNet[3]. ImageNet is a dataset created for object recognition, which does not match the type of photos typically taken by users. Places365, on the other hand, contains photos of many types of scenes, and might match the data better. For training, the same settings used before are used unchanged.

The reason why a *ResNet50* is used rather than one of the other ResNet variants, is because *ResNet50* was the only freely available pre-trained ResNet for the Places365 dataset. Although the deeper variants generally perform better, this is not an option for now. To allow fair comparison between Places365 and ImageNet, the model pre-trained on ImageNet is also a *ResNet50*.

[1] Normalization was done based on `https://pytorch.org/hub/pytorch_vision_resnet/`

[2] For details on why using SGD with momentum is a good idea, visit `https://towardsdatascience.com/stochastic-gradient-descent-with-momentum-a84097641a5d`

[3] The Pytorch *ResNet50* pre-trained on Places365 was downloaded from `https://github.com/CSAILVision/places365`

Model Name	Model Type	Pre-training Data	Data
Season1	ResNet50	Places365	All
EasySeason1	ResNet50	Places365	Easy Season
Season2	ResNet50	ImageNet	All
EasySeason2	ResNet50	ImageNet	Easy Season
Partofday1	ResNet50	Places365	All
EasyPartofday1	ResNet50	Places365	Easy Part of Day
Partofday2	ResNet50	ImageNet	All
EasyPartofday2	ResNet50	ImageNet	Easy Part of Day
AbeforeB	Siamese	Imagenet, Places365, Landmarks	All

Table 3.2: Overview of the different models for Season and Part of Day predictions.

3.2.3 Experiments

Both ResNets will be trained to determine which dataset is more suitable for pre-training. The following question can be formulated:

> **Q1.1**: *Of ImageNet and Places365, which is more suitable for the prediction of temporal information?*

The hypothesis is that the ResNet pre-trained on Places365 will lead to better performance. This question and the hypothesis do not just apply to Season prediction, but also to Part of Day prediction.

For a second experiment, a second dataset is created manually. The main problem with the original dataset is that the labels may be incorrect. These mistakes will cause the model to struggle with the data. Only images that seem to actually match their label are added to the smaller 'easy' dataset. For each of the classes, 150 images are selected. To prevent the model from only learning to classify snow as winter and the beach as summer, a serious attempt was made to not only include the most obvious photos. Photos that could belong to any season, or that have a clearly incorrect label, are not allowed in this dataset. The expectation is that this model at least does not make extreme mistakes, such as winter-summer mix-ups, when the visual content clearly indicates the actual season. However, the model might not perform as well as the model trained on all data, since there is only very little data available now. Overfitting could cause problems, and the model may not be able to generalize to more difficult data.

An overview of the different models for Season prediction is given in Table 3.2. For each of these models, predictions are visualized to analyze the mistakes made.

3.2.4 Results

This section is available upon request.

Hours	Part of Day
00-06	Night
06-12	Morning
12-18	Afternoon
18-24	Evening

Table 3.3: Part of day labels

3.3 Part of Day Prediction

3.3.1 Data

For Part of Day prediction, the hours are divided in 4 classes. The model is trained to predict 0 for *morning*, 1 for *afternoon*, 2 for *evening* and 3 for *night*. For Part of Day prediction, no further preprocessing of the data is required.

The labels are computed using the timestamps retrieved from the metadata. Each class has a duration of 6 hours, the distribution of the hours over the classes can be seen in Table 3.3. The exact hours chosen for each class are based on the fact that the morning ends at 12 PM. The data is again balanced to avoid overfitting. The balanced data contains, for a batch size of 32, 4153 batches for training, 1001 batches for validation and 625 batches for testing, equally spread across the four classes.

3.3.2 Training Procedure

The model used for Part of Day prediction is the same ResNet50 that is used for Season prediction. At first, the ResNet50 pre-trained on ImageNet is used. Again, the images are normalized[4] before feeding them to the ResNet. Since there are still four output classes, the same linear output layer is used as before. SGD is used, with the same parameters (0.01 learning rate, 0.9 momentum, 0.00001 weight decay). The learning rate scheduler is used with the exact same settings as before, and training is again stopped when there is no improvement for 15 epochs.

Next, the ResNet50 pre-trained on Places365 for Season prediction is also used for Part of Day prediction.

3.3.3 Experiments

Visual inspection of the data revealed many incorrect labels. The labels are mostly incorrect for users who have traveled across timezones with a device that did not automatically update to the local time. The most obvious mistakes are sunny photos taken around midnight, or the other way around. It is difficult to estimate how many images in the dataset have an incorrect label, since it seems impossible to find mistakes in photos taken just one or two timezones away from the user's home. One possible solution is to only train on photos taken with mobile phones, since these devices most often update automatically. However, this will result in a loss of data. Also, the dataset does not contain device information that can be used to filter photos taken

[4]Normalization was done based on `https://pytorch.org/hub/pytorch_vision_resnet/`

with smartphones. iPhones could also still cause a loss of metadata, as explained earlier.

So, rather than trying to filter out incorrect labels based on the device, a selection of data has been made. This selection does not contain obvious mistakes or photos that could have been taken during any part of the day. Similar to the Easy Season dataset described before, this easy Part of Day dataset also contains 150 photos for each class. The expectations are the same: the model will be less likely to make extreme mistakes, but might not be performing as well given that there is only little data to train on. Overfitting is a possible problem as well. Again, both ResNets are used to train on the easy dataset, resulting in four different models for Part of Day prediction. An overview of the different models for Part of Day prediction can be seen in Table 3.2.

3.3.4 Results

This section is available upon request.

3.4 A before B Prediction

3.4.1 Siamese Neural Network

To compare two photos on their relative temporal order, a Siamese Neural Network is explored. Recall that a Siamese Network allows for multiple inputs, and is able to determine some relationship between these inputs. While it is normally used to determine if the input images belong to the same class, it will now be used to determine which was taken earlier. Note that the label belonging to each pair depends on the order of the images inside that pair. If A is the first image and B the second, the label would be different from when A is the second image and B the first. This strategy defies one of the two key properties described by Koch et al. [2015], as the model is no longer symmetrical. For this model to be symmetrical, the output should be equal no matter the order of the two images.

Another question can be formulated for the issue of A before B prediction:

> **Q1.2**: *How well suitable is a Siamese Neural Network to solve the issue of A before B prediction?*

The hypothesis is that this network is well suitable, given that Siamese Neural Networks are specialized to compare images.

3.4.2 Data

The data is loaded as usual, but now chronologically ordered, and grouped per user. Next, successive images are paired. The data is paired per user since this will be the case for the end goal as well. Another reason for this is that the relative timestamps will be correct: even if the device has the wrong time stamps, it will not have messed up the order in which photos were taken (unless the photos were taken with different devices). The images within each pair are swapped at random to obtain an equal

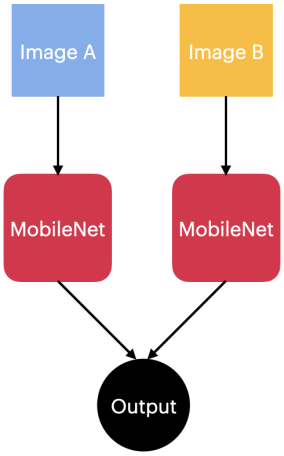


Figure 3.1: Siamese Neural Network for A before B prediction. Image A and Image B are each fed into an identical version of a MobileNet. The outputs are compared using a sigmoid activation function to obtain one final output value between 0 and 1, where 0 is $A \leq B$ and 1 is $B > A$.

number of A before B and B before A pairs. If the second image, B, is taken before the first image, A, the label is 1. Otherwise, the label is 0. So, if the images were somehow taken at the exact same moment, the label will be 0 as well.

3.4.3 Training Procedure

A MobileNet, pre-trained on ImageNet, Places365 and Landmarks, is used as base model for the Siamese Network. Feature vectors for both inputs are computed using the MobileNet. Next, the distance between these feature vectors is computed in a customized layer. The prediction output is obtained using a Dense layer with sigmoid activation. The Siamese Network is visualized in Figure 3.1.

The images are now normalized in range $[-1, 1]$, since this MobileNet was created using Tensorflow. The Binary Crossentropy function is used for the loss. The optimizer used is SGD, using a momentum and learning rate of 0.9 and 0.01 respectively.

Again, a learning rate scheduler is used, based on the same settings used for the learning rate scheduler for Season and Part of Day prediction. Again, the model keeps track of the validation loss during training and stops when there has been no improvement during 15 epochs. The model weights that are saved, are those obtained during the best epoch.

To avoid overfitting as much as possible, the data is balanced such that for each batch, half of the photos belong to one class, and the other half to the other. At first, each batch would first have only photos belonging to one class, and then only photos belonging to the other class. So, the images within a batch are again shuffled, just before being fed to the model.

3.4.4 Results

This section is available upon request.

Chapter 4

Image Based Time Synergy for Ordered Images

While the final goal is to solve the issue of event clustering for any set of randomly ordered photos, this thesis attempts to solve the issue for a chronologically ordered set of photos. This is an easier problem to solve, given that this orderering provides some temporal information. For example, when the set of photos is chronologically ordered, the algorithm cannot accidentally cluster photos into one event when they were taken relatively long apart. The probability of being able to solve the problem for unordered images seems dependent on the solvability of this slightly easier problem, making this a logical first step. Although the photos are chronologically ordered, no further temporal information is made available for the algorithm to cluster the photos into events. A realistic example of when a set of photos can be chronologically ordered while they have no temporal information present in the metadata, is when the filenames contain a number. Most devices have some structured format for image filenames.

Evaluation is done by using the hierarchical cluster tree obtained using the existing time-based approach as ground truth, and by visual inspection. The time-based event clustering is not the actual ground truth, given that this approach itself is not perfect. However, it is the best available set of event labels available. Visual inspection will reveal the gravity of the mistakes made.

The approach is largely inspired by Loui et al. [2005], using background and foreground segmentations. While Loui et al. [2005] used a block-based approach to compute foreground and background regions, here the *FgSegNet* created by Ang Lim and Yalim Keles [2018] will be used instead. The main reason to use this neural network rather than the original segmentation approach from Loui et al. [2005], is that the latter is only described in words, while the *FgSegNet* is publicly available. Also, the *FgSegNet* scored best on the challenge it was created for, so it can be assumed to be a suitable alternative[1]. The assumption is that it performs just as well or even better than the segmentation algorithm from Loui et al. [2005]. While their approach was

[1]The results for the change detection challenge, for which the *FgSegNet* was used, can be found on `http://changedetection.net/`. The *FgSegNet* scored best.

block-based, the *FgSegNet* is not restricted to classifying blocks, and is pixel-based instead. As explained by Loui et al. [2005], their algorithm would become too slow when using a pixel-based approach. One major advantage of the work of Loui et al. [2005] is that they compare different foreground regions separately, instead of treating all foreground pixels as one whole. Their approach results in a higher similarity when multiple different foreground regions from two different images match. Treating all foreground pixels as one region does not allow for these precise comparisons, which might lower the performance.

By combining the ideas of Loui et al. [2005] with the FgSegNet developed by Ang Lim and Yalim Keles [2018], we try to obtain event clusters based on detected event boundaries. The term *event cluster* is used to describe an event, consisting of one or more images. The term *event boundary* is used to describe the boundaries between successive event clusters.

4.1 Data

For event clustering, all images taken by one individual user are grouped together. Next, a lower limit of 50 images per user is used to ensure that there are enough events and sub-events to evaluate. Also, an upper limit of 125 is used, simply because a larger number of events takes too much time to evaluate. The priority is to evaluate the event clustering for multiple different users, rather than evaluating a few huge sets of photos. 25 sets of photos are used for evaluation, containing a total of 1990 images. These photos will be referred to as *User Photos*.

A second, very small dataset is used as well, which is a private set of holiday photos. These photos will be used in this thesis for visualization purposes. The dataset will be referred to as *My Holiday Photos* and consists of 160 photos from a 7-day trip to Poland. All timestamps for these photos are known to be correct, and the actual events that took place are known as well. Any people present in these photos (other than myself) will be covered by a colored square, for privacy reasons.

4.2 Foreground Segmentation

The model described in Section 2.3.1, the *FgSegNet*, will be used for foreground segmentation. Ang Lim and Yalim Keles [2018] made their code freely available online, and from the paper it seemed that they provided a fully trained model ready to be used on our own data.

However, it turned out that separate models were trained on each sub-category of the dataset, so there are actualy 53 models, each specialized on a single video feed. The images within each feed only have some varying objects moving through, such as a car or a person, while the main picture remains the same. Some small tests revealed that these different models may not work well enough on the User Photos. The models are too specialized on the video feeds and do not seem to generalize to different, unseen data. It remains unclear if Ang Lim and Yalim Keles [2018] actually used one model in their paper, or if they used these 53 specialized models.

4.2.1 Training Procedure

As a solution, the training process for *FgSegNet* was adapted. Rather than creating a new model for every video feed, one model is trained on all feeds sequentially. A batch size of 16 is used, and the model is trained for just one epoch on each video feed. This is done in a loop of 15, meaning that the model is trained 15 times for one epoch on each feed. Althoug not ideal, the resulting segmentation masks were much better than before.

Training on video feeds sequentially is expected to give some undesirable patterns. To avoid such patterns, a different approach was tried. For this approach, the images from the different feeds were all resized to the same size (the video feeds were of different sizes), so that they can easily be mixed for training. However, the model resulting from training on all 53 video feeds at once, resulted in a model that was not able to generalize at all and segmentation masks that were either entirely foreground or background. An attempt was made to find a more suitable foreground segmentation model, but none were made publicly available. It is simply not in the scope of this thesis to develop a segmentation model from scratch. A big issue here is that our user data does not have ground truth foreground masks to train on.

Once a trained model has been obtained, it is visually tested on user photos to ensure that it works well enough. These user photos do not have a ground truth for foreground segmentation, so no proper evaluation metric will be applied. The obtained masks will be used in the next step.

4.2.2 Results

Just to give an idea of the segmentation masks created by this model, a few examples are created using My Holiday Photos. These examples can be seen in Table 4.1. These examples were chosen to show the performance of the foreground segmentation model for different types of photos, such as landscapes, buildings and people. The model performs wel when there is a person present in the image. The photo of the horses shows that the foreground mask may not always have a perfect outline, just like the photo of the flowers. When there is no foreground available, such as in the photo of the mountains, the model either returns an empty mask, or a mask of something that is not actually the foreground. In this case, the model decided to use the clouds as foreground. Photos of buildings are also difficult for the model. Since the *FgSegNet* was trained for motion detection, buildings will never be detected as foreground, which is usually correct. In the photo shown in Table 4.1, however, the buildings could be considered to be foreground. The segmentation model decided to use parts of the surrounding buildings as foreground. Overal, the segmentation network seems to do a pretty good job, especially considering the problems with the training procedure and limited training data, and it can thus be used in the computation of image similarity.

Given that the foreground segmentations are only used to compare multiple images, it may still work when the segmentations are consistent over multiple pictures of the same building. For landscapes however, empty masks will be a problem: the computations for image similarity will then have to rely on the other features for image

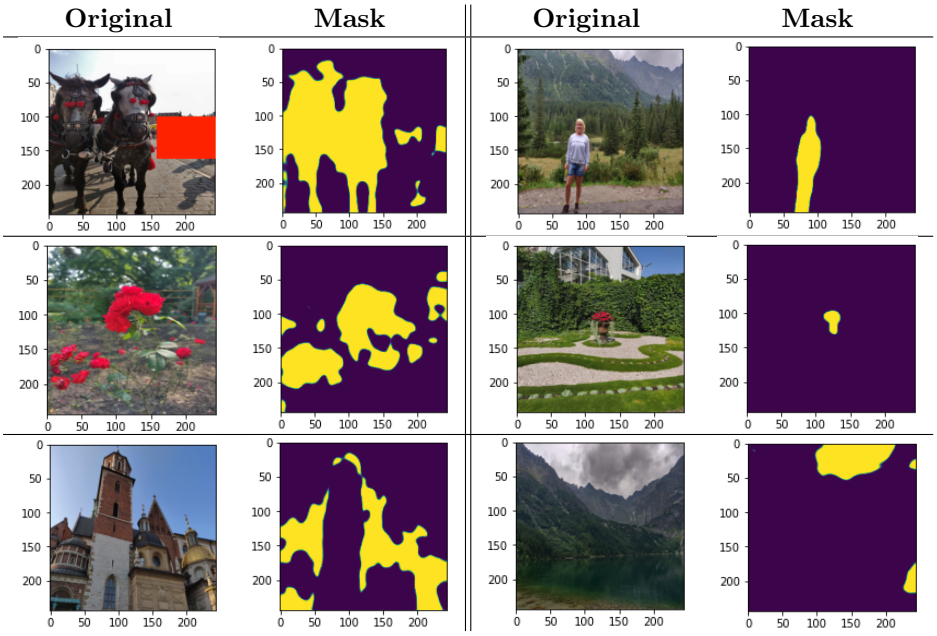


Table 4.1: Multiple example foreground segmentations for different types of photos. On the left are the original photos, on the right are the corresponding foreground masks. The original images are edited with a red square to remove people (other than myself) from the pictures, to avoid privacy issues. These photos were taken from My Holiday Photos.

similarity. To make sure that this goes well, the algorithm developed by Loui et al. [2005] needs to be adapted.

4.3 Computing Image Features

4.3.1 Features

Similar to Loui et al. [2005], the luminosity and a color feature is computed. They are comparing regions across images, and not within one image. However, they also seem to compare different "levels", i.e. possibly foreground and background regions as well. Their equations are not entirely clear, they might have wanted to keep them a little vague for the sake of their patent. In this thesis, a decision was made to try a simplified approach for the luminosity and color feature instead.

For this simplified approach, the luminosity and color features are computed over the entire image rather than per region. The luminosity is then simply the average luminosity of all pixels within an image. Equation 2.4 is used unchanged. For every pixel, y is computed using the R, G and B of that pixel. The resulting y is divided by 2.55 to obtain values in range $[0, 100]$ rather than $[0, 255]$. This range makes it easier later on to combine the different features into a final similarity score. After computing the y for every individual pixel, the average y is computed and saved as luminosity feature.

For the color feature, more extensive changes are made. Rather than using the equations provided by Loui et al. [2005], more general computations are used. Their computation for the hue is able to deal with imaginary numbers, which was very difficult to implement in Python, so the decision was made to follow a different computation instead. Instead of computing the hue, saturation and intensity, the hue, saturation and value (HSV) are computed according to Rapid Tables. For these computations, R, G and B are first divided by 255, putting their values in a range of $[0, 1]$. Some checks were done to make sure that the implementation of the new equation was correct. Again, the computations are done for the full image, rather than comparing the foreground and background in one image.

The new hue computation is given in Equation 4.2 and depends on the largest color value: R, G or B. To compute the hue, the difference between the maximum and minimum value needs to be computed first. This is shown in Equation 4.1, where $max = max(R, G, B)$ and $min = min(R, G, B)$. The output of the equation for hue is divided by 3.6 to obtain values in range $[0, 100]$ rather than $[0, 360]$.

$$df = max - min \tag{4.1}$$

$$Hue = \begin{cases} 0 & \text{if } df == 0 \\ \frac{60 \cdot (\frac{G-B}{df} \mod 6)}{3.6} & \text{if } max == R \\ \frac{60 \cdot (\frac{B-R}{df} \mod 2)}{3.6} & \text{if } max == G \\ \frac{60 \cdot (\frac{R-G}{df} \mod 4)}{3.6} & \text{if } max == B \end{cases} \tag{4.2}$$

The saturation also depends on max and df and is given in Equation 4.3. The output of this equation is in range $[0, 100]$.

$$Saturation = \begin{cases} 0 & \text{if } max == 0 \\ \frac{df}{max} \cdot 100\% & \text{otherwise} \end{cases} \tag{4.3}$$

The value only depends on max and is shown in Equation 4.4. The output of this equation is in range $[0, 100]$.

$$Value = max \cdot 100\% \tag{4.4}$$

The hue, saturation and value are averaged over all pixels, multiplied with a coefficient, and then summed to a total score. The equation is given in Equation 4.5. h, s and v are the coefficients for hue, saturation and value respectively. The values chosen for these coefficients are $h = 0.34, s = 0.33, v = 0.33$, so that the final *ColorFeature* value is in range $[0, 100]$ as well. A slight emphasis was put on the hue, based on a personal assumption that the color shade is more important than the saturation of value. The value is also partially covered by the luminosity. These coefficients are not optimized yet, which should be done in the future to obtain optimal results. The *hues*, *saturations* and *values* are the sets of hue, saturation and values for all pixels.

$$ColorFeature = h \cdot mean(hues) + s \cdot mean(saturations) + v \cdot mean(values) \quad (4.5)$$

As an experiment, Part of Day predictions and Color Histograms are also tested as features. Recall that the Part of Day is either 0 for morning, 1 for afternoon, 2 for evening and 3 for night. The model for Part of Day predictions is far from perfect, so including these predictions in event clustering could potentially lead to mistakes. Some measures will be taken to minimize the chance of mistakes being made, which is explained in Section 4.4. The Color Histograms are computed for only the backgrounds of the images, based on the hypothesis that the backgrounds within an event are similar across the photos from that event. This idea is more in line with the work of Loui et al. [2005], where regions of the same type (i.e. either foreground or background) were compared across multiple images as well.

To visualize the features, a set of three successive images is compared in a pairwise manner in Figure 4.1. The middle image (present in both Figure 4.1a and 4.1b) is compared to both the first and the third image, while the first and third image are only compared to the middle one, but not to each other. It should already be clear from the images that the second and third images should be clustered together, separately from the first image. The feature values for each of the images reveal that this would indeed happen: the values of the middle image are much more similar to the values of the third image than to the values of the first image.

4.4 Comparing Successive Images

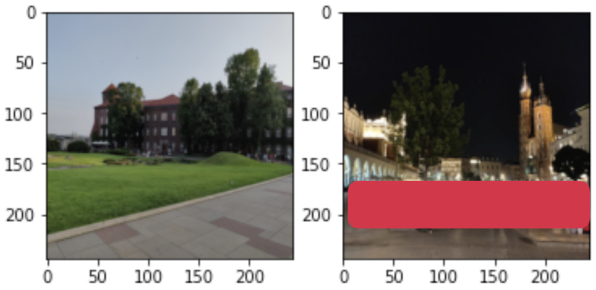
4.4.1 Similarity Features

Next, the similarity between successive images is computed. For each pair of images, multiple comparisons are made, before computing a mean similarity score. Firstly, the luminosity and color features for the full images are compared. Secondly, the foreground segmentation masks are compared using the Mean Squared Error (*MSE*) and Structural Similarity Index (*SSIM*)[2]. *SSIM* was developed by Wang et al. [2004] and is known to be better for image comparison than simply computing the *MSE*, the full equations can be found in their paper. In this thesis, the specific parameters for *SSIM* were set to match the original implementation[3]. Also, *multichannel* is set to *True*, to deal with our images.

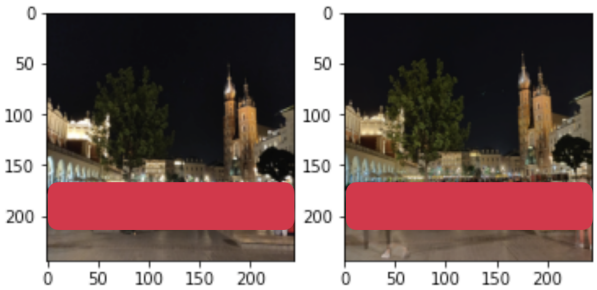
Given that the model for Part of Day prediction does not have a perfect performance, only an extreme difference between two images has a high chance of being reliable. If the prediction for the first image of a pair is two parts of day away from the prediction for the second image, the Part of Day difference is set to 0.5. Otherwise, a difference of 0 is used. So, when one image is presumably taken during the afternoon,

[2]The idea of using *MSE* and *SSIM* for image similarity was inspired by `https://www.pyimagesearch.com/2014/09/15/python-compare-two-images`

[3]*SSIM* parameters taken from `https://scikit-image.org/docs/dev/api/skimage.metrics.html#skimage.metrics.structural_similarity`



(a) Features for two dissimilar images.



(b) Features for two similar images.

Figure 4.1: Examples of values for the luminosity and color features, as well as Part of Day predictions. Again, red boxes are placed on people for privacy reasons.

Feature Name	**Equation**
$dist\_luminosity$	$abs(first\_luminosity - second\_luminosity)$
$dist\_color$	$abs(first\_color - second\_color)$
MSE	$MSE(first\_mask, second\_mask)$
$SSIM$	$SSIM(first\_mask, second\_mask)$
$ColHist$	$abs(first\_hist\_background, second\_hist\_background)$
$DayPart$	0.5 if $abs(first\_partofday - second\_partofday) == 2$, 0 otherwise

Table 4.2: Equations for succesive image comparison. Note that *first* refers to the first of two images, and *second* to the second image. *mask* refers to the corresponding foreground mask.

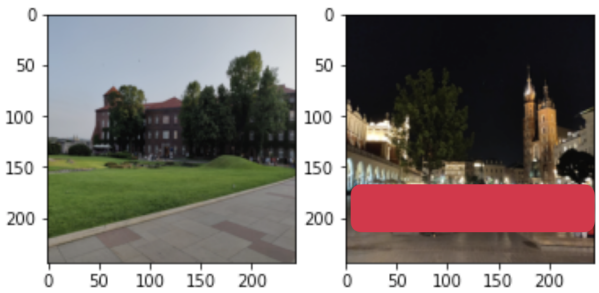
and another at night, the value of the feature is 0.5. If one image was taken during the morning, and the other during the afternoon, the value for the feature is 0. A maximum value of 0.5 was chosen, instead of 1.0, to make sure that this prediction does not have too much impact on the mean similarity. These two measures were used to prevent the algorithm from creating boundaries based solely on the possibly incorrect Part of Day predictions. The feature for Part of Day difference will also be referred to as *DayPart*.

The color histograms are used to compare the backgrounds of the two images. The Color Histogram differences are computed according to Rosebrock [2014], where 3D color histograms are created and normalized to finally be compared using the euclidean distance function. The feature of Color Histogram difference will also be referred to as *ColHist*.

Table 4.2 gives an overview of the features, where *first* and *second* refer to the first and second image. *mask* refers to the foreground mask of the corresponding image and *background* refers to the background obtained using the segmentation mask.

Although these similarity features were all placed in the same range ($[0, 100]$), they resulted in very different distributions in practice. While some features would often have values towards 1, others would hardly get close to 0.5. These differences lead to one feature having more impact on the mean similarity than other features. To solve this problem, several sets of photos were checked to determine logical maximum and minimum values for each feature. Then, every feature value is first subtracted by the minimum and then divided by its potential maximum. This normalization method gives us a normal-like distribution for each feature that is a lot easier to compare. The values would ideally lie between 0 and 1, but as this was a squewed distribution, some outliers will be outside of this range, resulting in values smaller than 0 or greater than 1. This is however not a problem for the computation of the similarity. The DayPart values are not adjusted and are still either 0 or 0.5.

Examples of the image similarities are visualized in Figure 4.2. Again, it is clear that the bottom two images are much more similar than the top two.



dist_luminosity : 0.70

dist_color: 0.20

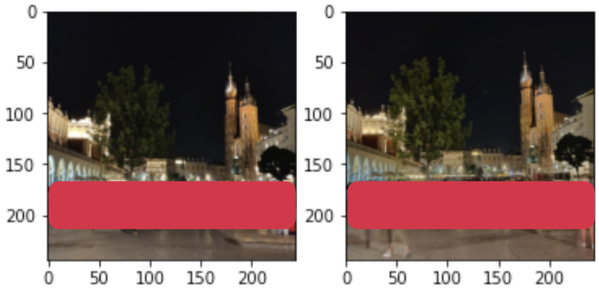
MSE: 0.11

SSIM: -0.08

ColHist: 0.36

DayPart: 0

(a) Similarities for two dissimilar images.



dist_luminosity : 0.08

dist_color: 0.06

MSE: 0.08

SSIM: -0.10

ColHist: -0.24

DayPart: 0

(b) Similarities for two similar images.

Figure 4.2: Examples of similarities between two images for all features. Again, red boxes are placed on people for privacy reasons.

4.4.2 Computing the Mean Similarity

Next, the mean similarity between each two images is computed. While Loui et al. [2005] computed the harmonic mean of the pairwise distances between regions in two successive images, this thesis explores different computations of the mean based on the equations from Table 4.2. The timewise computational complexity would have skyrocketed when comparing all separate foreground regions in a pairwise manner, since the segmentation model does not work well enough to obtain complete foreground regions. The model often returns masks with many separate pieces, which can be seen in the photo of the flowers in Table 4.1.

For each feature in Table 4.2, a higher value denotes a greater dissimilarity. Similarly, a higher mean similarity score also denotes a greater dissimilarity.

First, a harmonic mean is computed according to Equation 4.6. Next, a basic mean is computed according to Equation 4.7. Finally, a combined mean is computed as well, according to Equation 4.8. Each of these means will be computed to see what works best. The equations given here are based on the main set of features: *MSE*, *SSIM*, *dist_luminosity* and *dist_color*. The equations are extented to include DayPart and ColHist comparisons in some experiments. Given that the features are different from those used by Loui et al. [2005], and most likely in different ranges, Equation 4.7 seems the safest way to compute the mean similarity between two images.

Examples of mean similarities between images are given in Figure 4.3. All three means show a clear difference between the first and second pair.

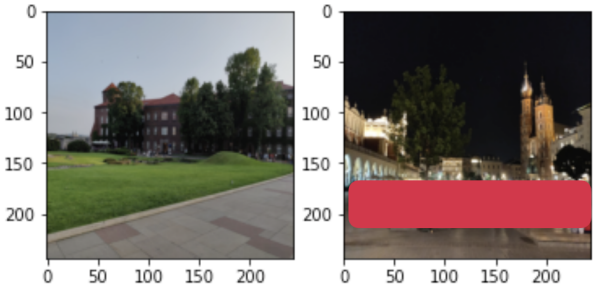
$$harmonic\ mean = \frac{1}{\frac{1}{SSIM} + \frac{1}{MSE} + \frac{1}{dist\_luminosity} + \frac{1}{dist\_color}} \tag{4.6}$$

$$mean = \frac{MSE + SSIM + dist\_luminosity + dist\_color}{4} \tag{4.7}$$

$$combined\ mean = \frac{harmonic\ mean + mean}{2} \tag{4.8}$$

4.5 Event Clustering

Loui et al. [2005] use one threshold for event clustering, either manually chosen, or computed using for example some function based on the average and maximum distances in a set of images. A disadvantage here is that there will be just one layer of events, disregarding sub-events. Another disadvantage is that this approach is invariant to local fluctuations in the similarity scores. As an alternative solution, the similarity values themselves are used to determine the event boundaries at multiple levels. The full set of event boundaries is detected first using the similarity between each two successive images. For every two images, a boundary between them is created if the similarity score is higher than for the previous two images. Next, these boundaries can be used to obtain the event clusters themselves, and to create image-based hierarchical trees. Once all event boundaries have been detected, the corresponding similarities are gathered in a list. To create a tree, this list is split in half based on

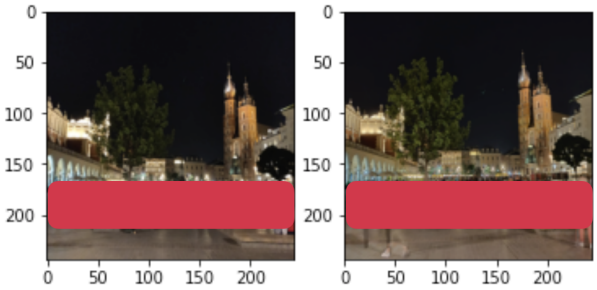


Mean: 0.22

Harmonic Mean: 0.20

Combined: 0.21

(a) Mean similarities for two dissimilar images.



Mean: -0.02

Harmonic Mean: 0.04

Combined: 0.01

(b) Mean similarities for two similar images.

Figure 4.3: Examples of mean similarities between two images. Again, red boxes are placed on people for privacy reasons.

the largest dissimilarity. This is repeated for both halves of the resulting lists until all boundaries have been added to the tree. The hierarchy in the tree is thus determined by the largest dissimilarities between two images. In this section, multiple experiments are carried out to determine which method works best to detect event boundaries.

4.5.1 Experiments

Similarity Features and Mean Similarity

As mentioned before, three computations for the mean similarity between images are tested: the mean (Eq. 4.7), harmonic mean (Eq. 4.6) and the combined mean (Eq. 4.8). Also, multiple sets of features are tested. The main set consists of *SSIM*, *MSE*, *dist_luminosity* and *dist_color*. For the experiments, *ColHist* and *DayPart* are added as well.

The different means and the different sets of features are tested on both My Holiday Photos and User Photos. First, the best mean calculation will be picked based on the performance on the User Photos. Then, the best set of features will be determined using the best mean calculation. The expectation is that the main set of features, together with the *ColHist* feature, will lead to the highest performance. Including *DayPart* as a feature is expected to lower the performance, given that the Part of Day model still makes many mistakes.

Applying a Threshold

Next, a threshold T is applied to the list of boundaries. Any detected boundary with a similarity in the lowest 1%, 5%, 10% or 15% of the similarities is removed. This could occasionally lead to a correctly detected boundary being removed, lowering the recall, but it should mostly remove wrongly detected boundaries from the set (increasing the precision). The experiments for the threshold are carried out using the best mean calculation and best set of features found before.

Boundary Levels

So far, the focus was to find a set of events and the corresponding event boundaries. This was done on one level, where every boundary is assumed to be equally important. Similar to the time-based hierarchical trees, an attempt is made to create an image-based hierarchical tree of event clusters as well. The importance of a boundary is determined by the degree of dissimilarity between the two images corresponding to this boundary, based on the assumption that events further away will have a larger dissimilarity in successive image comparisons. An image-based tree will be created and compared to the time-based tree for My Holiday Photos. Also, the boundaries detected by the image-based approach will be matched to the time-based tree, to determine how well the image-based method works for different levels of boundaries. Boundaries higher up in the time-based tree are more important than boundaries further down. These comparisons will be made for My Holiday Photos and two sets of User Photos. The following question can be formulated:

> **Q2.1**: *How well does image based time synergy detect higher levels of event boundaries?*

The hypothesis is that the algorithm is best at detecting the higher-level event boundaries. This is based on the assumption that photos from different main events will be less similar than photos from different sub-events within the same main event. However, it is expected that the algorithm works fine for lower-level boundaries as well, considering that the images within sub-events could still be very dissimilar.

Similarity Matrix

Finally, a first attempt to create and use a similarity matrix is made. This matrix compares every two images rather than just successive pairs, using the same similarity computation developed above. Using this matrix, a tree is created in the same way as in the time-based approach. This tree, unlike the hierarchical trees, will have just one level, where every boundary is equally important. This happens because, unlike the time-based tree, this tree only uses one similarity matrix. The set of boundaries will be extracted from this tree and compared to the time-based set of boundaries. This experiment will only be done for My Holiday Photos.

The following question is formulated:

> **Q2.2**: *How well does the all-to-all image comparison method work when compared to the successive image comparison method?*

While for a chronological ordered set of images, it makes sense that the highest similarities are around the diagonal, this may be very different for visual similarity. Photos from the first event in a set of photos may have a high similarity with photos from later events, so the similarity matrix is expected to be more chaotic. The resulting matrix-based tree is also expected to be different from both the time-based and image-based trees. However, the similarity matrix is expected to be less sensitive to the problem of one-off photos. In the end, the hypothesis is that the all-to-all method and the successive pairs method lead to similar performance.

Evaluation

To obtain ground truth event clusters, the time-based approach is used to create hierarchical event trees. From these trees, the boundaries are extracted. These boundaries are used to compute the precision, recall and F_1-score. The performance is computed over 25 sets of User Photos, and over My Holiday Photos.

Further evaluation of the resulting events and trees is done by visual inspection of at most three sets of photos.

4.5.2 Results

This section is available upon request.

Chapter 5

Discussion

5.1 Result Analysis

Foreground Segmentation

The FgSegNet was expected to have at least the same performance as the segmentation approach from Loui et al. [2005]. The pixel-based approach of this network was expected to be more precise than the block-based approach used by Loui et al. [2005]. However, no evaluation of the foreground segmentation method used by Loui et al. [2005] was provided. Their work only provides very few example segmentations, that seem to be accurate, but might not be representable for the total set.

It turned out that the *FgSegNet* used in this paper was perhaps too specialized to generalize to unseen data. The adapted training process was not ideal either, so the resulting foreground segmentations, such as those seen in Table 4.1, are far from perfect. However, they seem to cover most of the actual foreground, missing some and adding other pieces. Quantitative evaluation of the foreground segmentations is impossible, since no ground truth masks were available. It is clear that there is much room for improvement, but the segmentations are expected to be good enough for event clustering.

The rest of this section is available upon request.

5.2 Limitations

Incorrect Labels

For event clustering, it is important to include temporal information in the clustering process. As explained before, images from separate events can be very similar. So, if the predictions about the temporal information are not included in the computation of the similarity measure, this can result in incorrect event clusters. The idea was to include both Season and Part of Day predictions in the similarity measure, to make sure that this does not happen. However, reliable networks are crucial to make reliable predictions, and to train a reliable network, a reliable dataset is required.

During the prediction of temporal information, it became clear that the labels were not always correct. The model was thus trying to learn from a dataset with multiple incorrect datapoints, leading to peculiar predictions during testing. Training the model on easier datasets, where the labels were manually evaluated to at least be plausible, revealed that it would be possible for a neural network to predict temporal information based on visual content. This again stresses the importance of a reliable dataset.

The performances of the Season and Part of Day models were low, and so only Part of Day predictions were included for now. If a more reliable dataset was available, the prediction of temporal information could be more useful for event clustering.

Ground Truth Events

The ground truth used to evaluate the event clusters, is based on the time-based approach. This is not the actual ground truth, and the performance scores are thus not fully reliable. For a proper evaluation, a new dataset is required with actual annotations of the events. The most reliable annotators are the photographers themselves, since only they truly know what events took place. However, a few random annotators would also be able to give annotations to the dataset. It should be noted that the problem of event clustering is also more difficult for random annotators, since they obviously have no memory about the events someone else experienced.

Size of Photo Sets

A final limitation for the data used for event clustering is the size of the sets of photos. For evaluation purposes, the size of each set of photos was restricted to anywhere between 50 and 125 photos. Larger datasets would be too much work to evaluate and smaller sets may not contain enough different events. However, sets may contain hundreds or even thousands of photos, and it is just as important to evaluate the performance of the algorithm on these larger sets as well. Similarly, the algorithm should also be able to deal with photo sequences that contain a minimal number of photos.

Foreground Segmentations

Recall that the performance of the foreground segmentation model is limited, meaning that the resulting foreground masks are not entirely correct. The model is found to be best at detecting people. Other than that, the model tends to return noisy masks: some detected pixels are actually background pixels, and vice versa. These masks influence the computation of the features for image similarity.

Foreground segmentations of landscape photos are often close to empty, given that these photos generally do not have any foreground objects. This would not have been a problem for the goals of Ang Lim and Yalim Keles [2018], which was motion detection. For event clustering, however, this leads to similar masks for completely different landscapes. This is not incorrect, but comparing them will lead to a high similarity. If successive images show different landscapes, while belonging to separate events, they might be clustered into one event anyway. The *MSE* and *SSIM* for the

successive masks will be really low. The other similarity measures, *dist_luminosity*, *dist_color*, and *DayPart* are based on the full images and will still result in a higher dissimilarity for different landscapes, partially making up for the very similar masks. *ColHist* does not suffer from similar empty masks either, since it is based on the entire background. It is unclear if these similarity measures are always enough to solve the problem.

Evaluation

It is important to know the performance of each step in the algorithm. For Part of Day predictions, the performance is known, but for the segmentation model, no scores are available. The foreground segmentations should be evaluated as well, but there are no ground truth masks available at this point. These ground truth masks should be created by some human annotators, but this is again a very time-consuming process.

As mentioned before, for proper evaluation of the entire algorithm, actual ground truth event clusters are required. Also, the visual inspection in this thesis is limited to just three sets of photos. Visual inspection should be extended to include many more photo sequences to get a decent sense of the performance of the algorithm.

Ordered Images

The algorithm developed in this thesis is only able to work with chronologically ordered photo sequences. The problem is much more difficult when the algorithm also needs to order the photos or the resulting event clusters. In reality, it cannot be assumed that the images in a series of photos are always ordered, so the algorithm should be extended to work with any unordered set of photos as well.

The rest of section is available upon request.

5.3 Future Work

Foreground Segmentation

Looking at the approach for foreground segmentation, there is a lot of room for improvement. The current model is trained on just 53 different video feeds and was originally created for motion detection. The resulting foreground masks for the user photos often include some random pixels classified as foreground. Replacing the currently used segmentation model with one that is better at generalizing to this data would most likely improve the performance of the event clustering algorithm.

To be able to quantitatively determine the performance of the current segmentation network (and its possible replacement), ground truth segmentation masks for the User Photos are required. Creating these masks is a huge task, but it would definitely be useful to see how well the segmentation model generalizes to out data. Or perhaps, this dataset could also be used to train the segmentation model in the first place.

It would be interesting to explore a different computation of the mean similarity, based on the size of the foreground segmentations. For images with no foreground, or

just a very small foreground, it may be better to compute the mean similarity without comparing the segmentation masks. Another possible solution, as implemented by Loui et al. [2005], is to simply use a square in the middle of the image as foreground, if no actual foreground is detected. Although this approach leads to incorrect foreground segmentations, they found a slight improvement in performance. Both ideas could be explored and might be able to solve the issue of comparing very similar empty masks.

Finally, Loui et al. [2005] were using a technique to obtain separate foreground regions for separate foreground objects. These separate regions were all compared individually to obtain a similarity score. In this thesis, all foreground pixels were treated as one whole. It would be interesting to see if the performance increases when implementing an algorithm to retrieve separate foreground regions and using the individual regions for comparison as well. But, as stated before, the timewise complexity is expected to explode for masks with many separate pieces.

Image Features

The features used for both individual images and the similarity between two images, are different from those used by Loui et al. [2005]. It still may be a good idea to try to follow their approach more strictly. However, this may be very difficult, given that not all specifics of their method are described extensively. An example feature that was not implemented in this thesis is the size of the foreground regions. This still seems like an interesting feature for image similarity that should be tested. This size-based similarity feature can be computed using a distance metric that compares the total number of foreground pixels detected in both images.

In this thesis, several features for image comparisons were used, although there are probably still many other similarity measures that could be explored as well.

Image Based Time Synergy for Unordered Images

This thesis has addressed the problem of image based time synergy for chronologically ordered sets of photos. The most important step for future work is to translate the algorithm developed in this thesis to also solve the problem of image based time synergy for unordered images. Season prediction and Part of Day prediction could be useful, although these models first need to be trained on a reliable dataset. A before B predictions could also be used here.

The rest of section is available upon request.

Chapter 6

Conclusion

This chapter is available upon request.

Glossary

burst Large set of photos taken in a small interval of time. 6

color feature A feature computed using the hue, saturation and value of a picture. 26

event Something that happens within a time interval, such as a birthday party, Christmas dinner, or a vacation. 1

event boundary A boundary between two successive events. In a set of pictures, the boundary is between two pictures, where the first belongs to the event before the boundary, and the second belongs to the event after the boundary. 6

hue The dominant color of a pixel. In this thesis, the average hue of a picture is used. 12

intensity The lightness, or value, of the hue of a pixel. 12

iPhone Apple smartphone. 1

luminosity The brightness of a picture. 7

saturation The strength of the hue of a pixel. In this thesis, the average saturation of a picture is used. 12

sub-burst Smaller peak of the number of photos taken during a smaller part of the time interval of a burst. 6

sub-event Smaller part of an event, such as eating dessert during a Christmas dinner. 6

value The lightness, or intensity, of the hue of a pixel. In this thesis, the average value of a picture is used. 27

visual content Anything that can be found in the pixel values of a picture. 2

visual similarity Similarity between two pictures, based on their visual content. 6

References

L. Ang Lim and H. Yalim Keles. Foreground segmentation using a triplet convolutional neural network for multiscale feature encoding. *arXiv*, pages arXiv–1801, 2018.

C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

M. Cooper, J. Foote, A. Girgensohn, and L. Wilcox. Temporal event clustering for digital photo collections. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 1(3):269–288, 2005.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

A. Graham, H. Garcia-Molina, A. Paepcke, and T. Winograd. Time as essence for photo browsing through personal digital libraries. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*, pages 326–335, 2002.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016a.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016b.

A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

A. Jaimes, A. B. Benitez, S.-F. Chang, and A. C. Loui. Discovering recurrent visual semantics in consumer photographs. In *Proceedings 2000 International Conference on Image Processing (Cat. No. 00CH37101)*, volume 3, pages 528–531. IEEE, 2000.

G. Koch, R. Zemel, and R. Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

A. C. Loui, M. Jeanson, and Z. Sun. Event clustering of images using foreground/background segmentation, July 5 2005. US Patent 6,915,011.

H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017.

Rapid Tables. Rgb to hsv color conversion. URL `https://www.rapidtables.com/convert/color/rgb-to-hsv.html`.

K. Rodden. *Evaluating similarity-based visualisations as interfaces for image browsing.* PhD thesis, Citeseer, 2002.

K. Rodden and K. R. Wood. How do people manage their digital photographs? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 409–416, 2003.

A. Rosebrock. How-to: 3 ways to compare histograms using opencv and python, 2014. URL `https://www.pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/`.

Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar. Cdnet 2014: An expanded change detection benchmark dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 387–394, 2014.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.