Machine Learning Based Video Editing Toolbox for Automatic Summary of Medical Videos

Emil Nilén, Olle Oswald

Master's thesis
2019:E25

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Centrum Scientiarum Mathematicarum

Abstract

Recording of medical procedures makes it possible for medical staff to review their work, learn from their mistakes and produce material for education. Medical videos tend to be long, which has an impact on usability and raises issues concerning memory storage.

The aim of this master thesis is to make the material more user-friendly with an intelligent toolbox based on machine learning and image analysis techniques. The tools divide the video into chapters with a k-means++ clustering technique, detect when an X-ray source is active with ROI based processing, track camera movements by comparing frames with the optical flow algorithm by Gunnar Farnebäck and identify when medical instruments are present using an artificial neural network. Based on the information from these tools a combined timeline and an automatic summary is created.

The results indicate that the chapter tool is especially promising when the videos include sections from before and after a medical procedure, since these are easier to separate. The region of interest (ROI) based tool detects all the frames with an active X-ray. The neural network performs well on classifying frames containing an instrument, but requires annotated data for training. The majority of camera movements are found, but the algorithm sometimes fails to detect zoom in the video.

This thesis is intended as a proof of concept of the potential in automatic processing of medical videos. The tools can create reference points to important sequences. More data and evaluation of the tools are necessary for the further development of an automatic summary system.

Acknowledgements

First and foremost, we would like to thank our supervisor Mikael Nilsson from the Department of Mathematics at LTH for his support and guidance throughout the process of this thesis. We would also like to thank Kalle Åström for accepting the role as our examiner.

Our collaborators Medical Imaging Technologies have embraced us with open arms and especially Per Wilhelmsson has been a great source of inspiration. We are thankful for the equipment, data and support they have provided us with.

Thank you also Kiet Tran, for giving us a clinician's perspective on the subject.

Last but not least, we would like to extend our immense gratitude to our friends and families for putting up with us during the course of this project.

Contents

1 Introduction

1.1 Background

An increasing amount of health data is overwhelming the healthcare system. There are many sources generating this data: images, physicians notes, readings from sensors and devices, etc. With this increase, a problem arises of how to handle all the data, which is too valuable to throw away. Possible uses range from diagnosing diseases in early stages, to predicting if a patient is at risk for surgery complications or hospital-acquired illness [1].

One segment of the data is video recordings. Per Wilhelmsson, Chief Technology Officer at Medical Imaging Technologies (collaborating company of this thesis), has 10 years of expertise from the surveillance industry, expressed the following thoughts on the future of medical video during an interview [2]: "The state of medical video recording today is in many ways similar to where the surveillance industry was in its beginning in the 80s, and is likely to follow the same journey. In the beginning, users were satisfied by simply having the equipment and no particular thought was paid to how to make use of all the video. As the technology became more common and widespread, people started to think about why they wanted to record everything, and what to actually do with the abundance of information that was created every day."

Medical Imaging Technologies [3] was founded in 2009, and provides solutions for audiovisual integration in hospitals worldwide. With over 100 installations in the Nordics, India, the UK and the United Arab Emirates, their technology enable medical professionals to record and share audio, image and video from the operation room to within the hospital or beyond. The reasons for wanting to capture such material may be e.g. to access remote expertise, distribute audiovisual content to a conference room or to hold an online master class of a procedure for pedagogical purposes. Apart from live streaming of medical audiovisual content, there is also an incentive for recording procedures. The reasons may be juridical – sometimes procedures must be recorded and saved for insurance purposes [2]. There are also educational possibilities, such as being able to watch previously recorded medical procedures, or to "self-review" after performing a procedure and answer such questions as "Did I make the right decision?" or "How do I make sure this does not happen again?" [4].

There is a gap between recording a medical procedure and actually making good use of the video. To sift through hours of raw video, where a lot of the time nothing interesting is happening, is time consuming manual labour. A summarized video decreases the amount of data and makes it manageable to watch. To obtain a summarized video of a medical procedure, a method is to do it manually which is also time consuming for a clinician. Many companies offer this service today where they have staff with medical expertise editing the videos [5] [6] [7]. By creating an algorithm that automatically creates highlights from the video, the need for manual editing could be removed which would save time and money. This is what Medical Imaging Technologies has realized and where this thesis comes into the picture.

1.2 Scope

The purpose of this master thesis is to investigate and adapt image analysis and machine learning techniques that can be used to develop "intelligent tools" in a video editing toolbox. The medical videos of interest for this thesis have been captured by Medical Imaging Technologies' InVision system during percutaneous heart procedures. A grid view example of the visual material is shown in figure 8 in the Data section. The main goal is to create a set of tools which can provide helpful information on a video editing timeline. A secondary goal is to create automatic summaries of the videos by combining the information from the tools, to save time for the user. In this work, four tools will be constructed. An overview of the tools and the techniques used to implement them is displayed in figure 1. The tools are further explained in the Method section.

The first tool builds upon unsupervised machine learning clustering techniques. Its purpose is to automatically divide the video into chapters which are coherent in time but also visually similar within each chapter.

The second tool utilizes a feature based method of recognizing when an X-ray symbol is present in the X-ray video. The purpose of this tool is to identify when the X-ray is active. This is motivated by the assumption that X-ray is used as sparingly as possible, and is only activated during critical episodes of a procedure.

The third tool is also a feature based method which uses a supervised classifier to identify when a medical instrument is present. This shows the possibility to customize a medical video summary depending on what objects are of interest to the user.

The fourth tool uses optical flow to detect when the camera is zooming in or being moved. Such manoeuvring of the camera can indicate that something interesting is happening during the procedure which the operator wish to capture in greater detail. This is a shot selection based method.

The tools will be combined to create an enhanced video timeline, allowing the operator to navigate through the material with more ease. For example, the automatic chapters (tool 1) brings some temporal structure to the video, while the X-ray identifier (tool 2) helps the operator to find sequences where there is X-ray related activity.

Finally, a short summary of a medical video could be automatically created using the tools, with the goal to retain as much information as possible while not allowing the summary to become too long.

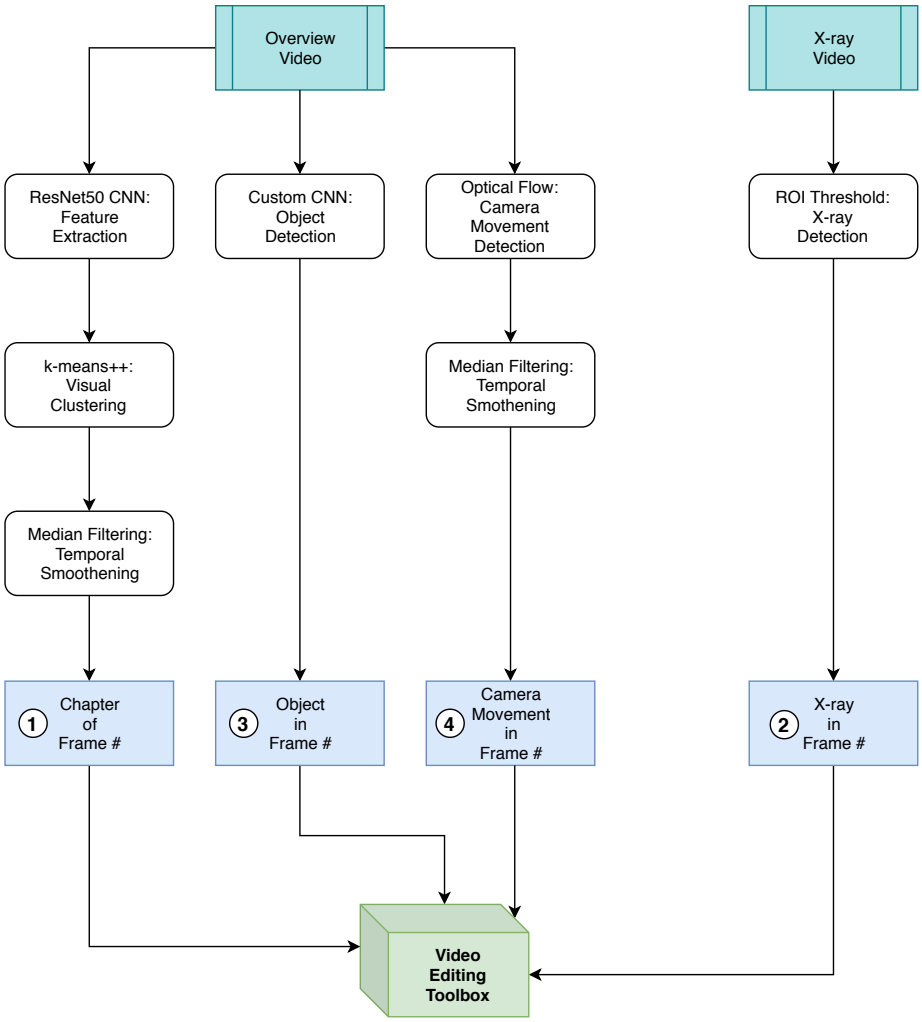

Figure 1: Overview of the tools and the techniques used to implement them. The numbering of each tool corresponds to the order in which they are treated throughout this report.

2 Theory

In this section, theory related to the development of the four tools is introduced. The theory covers previous approaches to video summary and the image processing techniques optical flow and region of interest processing. Machine learning concepts which have been used in this thesis are briefly covered, although some previous knowledge on artificial neural networks is likely to aid the reader in better understanding the contents.

2.1 Previous work on automatic video summary

As the amount of video content in the world increases rapidly, the need for solutions that enable effective storage and retrieval of relevant information has led to an emerging field of research and numerous suggested solutions to the problem of video summarization. The suggested approaches each have advantages and disadvantages and are more or less suited to the particular characteristics of a certain video or genre of videos. A broad classification of these techniques into six categories, some of which can be further divided into subcategories, has been suggested by [8], where the suitability of each technique for different genres and video characteristics is also discussed. The six main categories of summarization techniques are:

- **Feature based:** Feature based summarization techniques build upon detecting certain features of a video, such as color, gestures, audio, motion or certain objects. The detection of these features, or changes to them, are then used to select keyframes which, hopefully, can give a meaningful summary of the video content.
- **Cluster based:** Clustering techniques attempt to group video frames together based on the similarity of some characteristic. Once the video has been clustered, keyframes may be selected from each cluster, or entire clusters may be concatenated or omitted based on some criteria to generate a summary of the video.
- **Event based:** The difference between a frame and a reference frame is calculated. If the difference is large enough a new event is said to have begun, and a new reference frame is chosen to calculate similarity of the subsequent frames to the new event.
- **Shot selection based:** A shot in a video is a set of frames which belong together in time and appearance. A movie scene where the camera shifts from one actor to another for example, consists of two shots. The detection of shots may be done by measuring the distance between frames using e.g. color histograms. After detecting shots, the user may then select the most important ones. These techniques are applicable only to videos captured by a moving camera.
- **Trajectory based:** In trajectory based approaches, a static camera such as a surveillance camera is assumed. A summary is then based on the behaviour of moving objects in the static scene, either by showing their trajectory over time or by only selecting frames where the trajectory of a moving object changes significantly.

- **Mosaic based:** These techniques stitch together consecutive frames to form a panoramic image. Assuming the background is static (the camera may move, however), a background mosaic can first be constructed. By analyzing the trajectory of moving objects in the foreground additional mosaics can then be added on top of the background.

2.2 Supervised machine learning

The choice of a machine learning algorithm is affected by what data exists for training and evaluation of the model. Depending on the available data, different techniques and algorithms are used to solve regression or classification problems. A common situation is to have data with labels to train the algorithm, which is the definition of supervised learning. Some common supervised algorithms are listed in table 1. A dataset can consist of a thousand images of dogs and the same number of images of cats, where every image has a label referring to if it is a dog or a cat. The dataset is split into two parts. One part is used to train the model to learn how to classify data, and the other part is for evaluation of the model. With the performance result from the evaluation, the model can be adjusted and optimized to increase the performance [9].

Supervised algorithms	
Classification	**Regression**
Logistic regression	Linear regression
Classification trees	Decision trees
Support vector machine	Bayesian networks
Random forests	Fuzzy classification
Artificial neural networks	Artificial neural networks

Table 1: A collection of algorithms used when the dataset consists of training data with ground truth [9].

2.2.1 K-fold cross-validation

K-fold cross-validation is a robust method to estimate the performance of a model. The method randomly splits the dataset in a number of folds, where one fold is used for evaluation and the rest are used to train the algorithm. K iterations are executed and for every iteration a new fold is used as test data. The performance result of the algorithm is the average of all the iterations [10]. Figure 2 shows the structure of a 7-fold cross-validation.

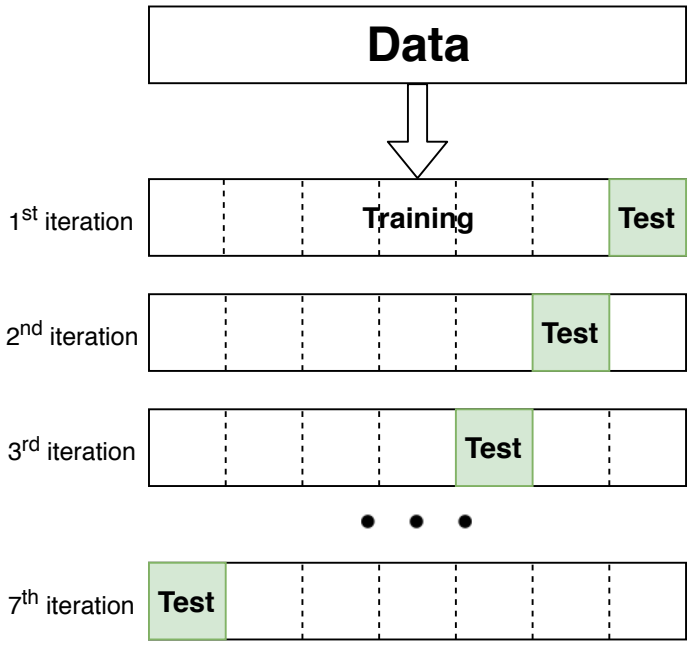

Figure 2: A 7-fold cross-validation where the green squares represent the data for testing and the white squares the data for training.

2.2.2 ROC analysis

Classifier models map data to predicted class labels. A discrete classifier outputs a class label directly, whilst a continuos classifier produces a continuous output. To predict class membership, various thresholds can be applied on the continuous output. A method to evaluate the classifiers is a receiver operating characteristics (ROC) analysis [11]. The ROC analysis provides information on the impact of a specific threshold. This is important to know when desiring a conservative classifier with low correct positive prediction rate (hit rate) and a false positive prediction rate (false alarm rate) close to zero, or a classifier with higher hit rate which allows the false alarm rate to increase. The true positive and false positive rates are explained in equations 1 and 2.

$$\text{True positive rate} = \frac{\text{Positives correctly classified}}{\text{Total positives}} \tag{1}$$

$$\text{False positive rate} = \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}} \tag{2}$$

A ROC graph visualizes the relationship between the true positive rate and the false positive rate. Varying the thresholds for a continuos classifier will result in different dots in the graph. A threshold at $+\infty$ corresponds to a dot at the coordinate (0,0) and $-\infty$ at (1,1), see figure 3. The representation of a discrete classifier is similar to a continuous classifier with one threshold. The aim is to find a classifier with a dot at (0,1) where all the positives are correctly classified and the false alarm rate is zero.

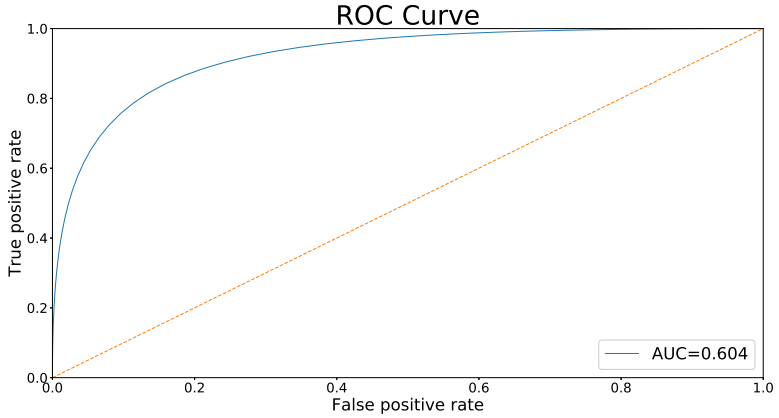

Figure 3: The ROC-curve visualizes the performance of a classifier. The blue line shows how the ratio changes with different thresholds. The dotted orange line corresponds to a randomly guessing classifier.

Area under curve (AUC)

Drawing a line from coordinate (0,0) to (1,1) via the classifier dot(s) enables calculation of the area under curve, which is a way of representing the ROC performance with a scalar. The randomly guessing classifier will have an AUC of 0.5, therefore a good classifier needs to perform above that, and preferably close to the optimal AUC at 1.0. The importance of AUC can be questioned as it is possible to have a classifier with a low AUC performing better than a high AUC classifier in certain regions, depending on the aim of the classifier.

2.3 Unsupervised machine learning

Without ground truth, it is difficult for an algorithm to learn from training data how to map data to labels. Instead, an algorithm has to focus on how data differs from, or resembles, other data, i.e. to find structures and patterns in the dataset. Classification of data can be performed by clustering on some data features, for example colours or shapes [9]. Some common unsupervised algorithms are listed in table 2.

Unsupervised algorithms	
Clustering	**Dimension reduction**
K-means clustering	Principal component analysis
Hierarchical clustering	Tensor decomposition
Gaussian mixture models	Multidimensional statistics
Genetic algorithms	Random projection
Artificial neural networks	Artificial neural networks

Table 2: A collection of algorithms used when the dataset does not consist of training data with ground truth [9].

2.3.1 Clustering

Clustering of data is about dividing samples into groups based on the observable information. There exists many algorithms which use different approaches for this cause.

Hierarchical clustering [12] is a collective name for clustering algorithms which divide data into sequentially smaller partitions, creating a tree-like data structure. At the root of the tree is a very large, single cluster containing all the samples. The root is repeatedly divided into smaller clusters, so that each leaf of the tree eventually contain only a single sample. Various metrics may be used for partitioning, such as average linkage, single linkage or Ward distance. Hierarchical clustering algorithms can be used for large quantities of data, but suffer from a problem of favoring clusters which are already large, sometimes referred to as "rich gets richer" [13].

DBSCAN [14] is a clustering algorithm which defines a cluster as a set of core samples. A core sample is a sample which has at least a *minimal number* of neighbouring samples within a distance ϵ. All core samples which are connected this way make up one cluster. In addition to the core samples, samples which are within ϵ of a core sample but are not themselves core samples are also part of the cluster. These non-core samples are typically found along the edge of a cluster. Samples which are further away than ϵ from any core sample are considered as outliers. DBSCAN is good for partitioning clusters of odd shape and different sizes, but the minimal number of neighbouring samples and the distance ϵ are parameters which must be carefully selected for the algorithm to work well [13].

The previously mentioned clustering algorithms use distance between samples as basis for partitioning data. The order in which data has been observed and the possibility of temporal dependencies is not taken into consideration, however. Hidden Markov Models (HMM) [15] are a family of statistical models which can be used for clustering of time-series [16]. They consist of a finite number of hidden states, which have a probability of generating certain outputs, known as the emission probability. In the model, there are also probabilites for transitioning from one state to another, known as transition probabilities. One problem formulation of HMMs is to estimate the underlying sequence of hidden states, which has generated a sequence of observations. HMMs have been successfully used in many applications such as speech recognition, financial predictions and time-series clustering, but require training data for estimation of parameters. Thus, HMMs are not suitable for entirely unsupervised clustering.

k-means clustering [17], also referred to as Lloyds algorithm, is an unsupervised clustering algorithm which has been called "one of the most popular approaches in the field of datamining" [18]. An improved version of this algorithm called k-means++ [19] has been selected for use in this thesis, due to its simplicity and scalability to large quantities of data, despite the fact that it does not take temporal order into account.

2.3.2 k-means++ clustering

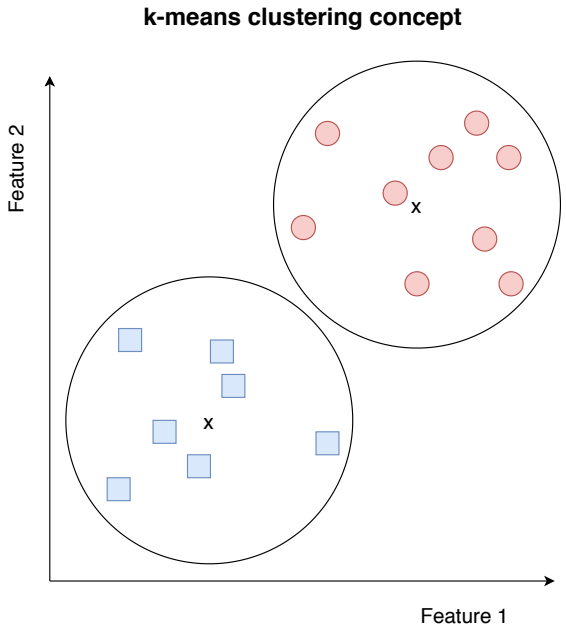

Figure 4: An illustration of the k-means clustering principle. The dataset has been divided into two clusters, based on the similarity between the samples w.r.t. the features "Feature 1" and "Feature 2". The "x":es symbolize each cluster center.

The basic principle of k-means is that all the samples in a dataset are assigned to k different groups (clusters), based on how similar they are with regard to some features. The similarity is usually measured by euclidean distance, and the goal of the k-means algorithm is to minimize the distance between each sample and the center of the cluster to which it has been assigned [17]. A simple illustration is shown in figure 4. With k as the number of clusters, x_i as the i:th sample and c_j as the j:th cluster center, the basic k-means algorithm can be described as

1. Select k cluster centers c_j at random
2. For a sample x_i, calculate the distance $||x_i - c_j||$ to each cluster center c_j and assign the sample to the nearest cluster
3. Calculate the centroid of each cluster (the mean of all its assigned samples) and move the cluster center there
4. Repeat step 2-3 until convergence (no reassignments of x_i) or for a given number of iterations

The speed and accuracy of k-means are dependent on the initial locations of the cluster centers, which in the basic algorithm are seeded uniformly at random in the feature space, or uniformly from among the samples. An extension to the basic k-means algorithm called k-means++ [19] uses a more advanced scheme

for seeding the initial cluster centers. k-means++ takes into account the squared distance from each sample to the closest cluster center which has already been chosen. A sample is more likely to be chosen as an initial cluster center if it is far away from an already existing cluster center. After the initial seeding of the cluster centers, k-means++ proceeds by (re)assigning samples to clusters and updating the cluster centers just as the standard k-means algorithm. With $D(x_i)$ being the distance between sample x_i and its nearest cluster center, the k-means++ algorithm looks like this

1. Select one cluster center uniformly at random from all the samples
2. Add a new cluster center from the samples with probability $\frac{D(x_i)^2}{\sum_i D(x_i)^2}$
3. Repeat step 2 until k initial cluster centers have been selected
4. Continue with the standard k-means algorithm

The probability $\frac{D(x_i)^2}{\sum_i D(x_i)^2}$ means that the further away a sample is from its closest cluster center (which implicitly means it is even further away from any other cluster center), the more likely it is to be selected as a new cluster center. The benefit is that the initial cluster centers will tend to be more spread out than if they had been seeded randomly. k-means++ has been shown both theoretically and empirically to outperform the standard k-means algorithm with regards to speed and consistency.

2.3.3 Clustering metrics

There are several metrics for evaluating the performance of clustering algorithms. What they have in common is that they quantify how well the data has been separated into different groups. In absence of a ground truth where the true "optimal" grouping is known beforehand, the unsupervised metrics are restricted to quantifying how similar or close the data are within a cluster or how well separated the clusters are from each other. Three such unsupervised clustering metrics are the Silhouette [20], Calinski-Harabaz [21] and Davis-Bouldin [22] scores. Clustering metrics can serve as guidance in determining how many clusters the data should be partitioned into, e.g. selecting a good value of k in the k-means++ algorithm [23].

Silhouette score

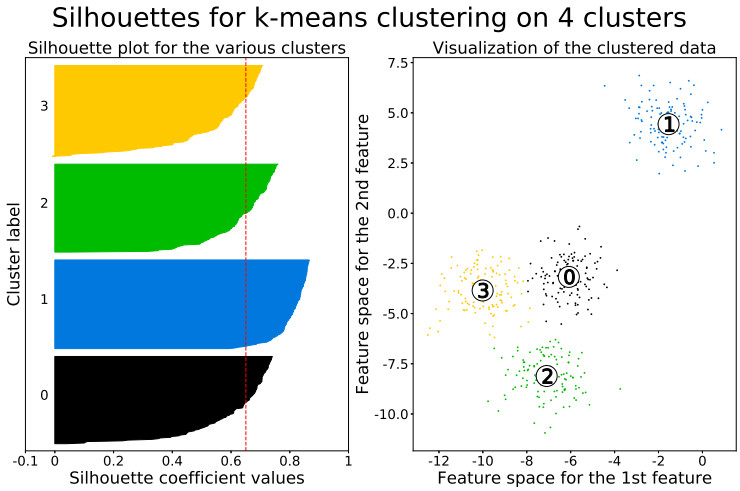

Figure 5: The cluster silhouettes for k-means clustering of sample data where $k = 4$. The width of a silhouette corresponds to the silhouette coefficient of individual samples, the height corresponds to the number of samples within a cluster.

Cluster silhouettes can be used for both graphical and numerical evaluation of the clustering of a dataset. The silhouette coefficient of a sample s, belonging to cluster A, is defined as

$$S_{silhouette}(s) = \frac{b - a}{\max(a, b)} \tag{3}$$

where a is the mean distance between s and all other samples within A, and b is the mean distance between s and all samples in *the nearest neighbouring cluster* B. Had s not belonged to A, it would likely have ended up in B instead. The value of $S_{silhouette}$ is bounded between $-1 \leq S \leq 1$, and says something about how appropriate the current cluster assignment of s is

- S is close to 1 implies that b is much larger than a. Sample s is likely in the correct cluster, since the average distance to B is much larger than the average distance to A.
- S is around 0 implies that a and b are approximately equal. Sample s is equally far away from both A and B, and therefore could belong to either cluster.
- S is close to -1 implies that a is much larger than b. s is likely in the wrong cluster, as it is closer on average to B than to its own cluster A.

All the individual silhouette coefficients can be combined to create a silhouette plot, as shown in figure 5. In a silhouette plot, the width of a cluster silhouette

corresponds to the silhouette coefficient of individual samples and the height corresponds to how many samples are within each cluster. Wider silhouettes indicate better pronounced clusters.

The individual silhouette coefficients can also be used to calculate an overall average silhouette width $\bar{S}$, which is the average of the silhouette coefficients over the entire dataset. This single number can be used as guidance to determine an optimal number of clusters, by selecting the value of k which yields the highest $\bar{S}$ [20].

Calinski-Harabaz index

The Calinski-Harabaz index compares the ratio between "within group sum of squares" and "between group sum of squares". It is often referred to as the Variance Ratio Criterion, as it describes the ratio of the intracluster variance and the intercluster variance. Mathematically, the Calinski-Harabaz index is given by

$$S_{CH}(k) = \frac{Tr(B_k)}{Tr(W_k)} \frac{N-k}{k-1} \tag{4}$$

$$W_k = \sum_{q-1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \tag{5}$$

$$B_k = \sum_{q} n_q (c_q - c)(c_q - c)^T \tag{6}$$

where $S_{CH}(k)$ is the variance ratio, k is the number of clusters, N is the total number of samples, C_q is the set of samples in cluster q and c_q is the cluster center of cluster q. $Tr(A)$ is the trace of a matrix A. The variable c is the center of all the cluster centers. It is desirable with compact clusters which are well separated from each other. This translates to a low intracluster variance ($Tr(W_k)$) and a high intercluster variance ($Tr(B_k)$). Thus, larger values of $S_{CH}(k)$ indicate a better choice of the number of clusters k [21] [23].

Davis-Bouldin

The Davis-Bouldin index is a similarity measurement describing the average similarity between each cluster and its nearest neighbour. The similarity in the Davis-Bouldin index is defined as

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{7}$$

with s_i being the cluster radius, i.e. average distance between each point in cluster i and its centroid, and d_{ij} being the distance between the centroid of cluster i and cluster j. The Davis-Bouldin index is defined as the average similarity between each cluster and its nearest neighbour

$$S_{DB}(k) = \frac{1}{k} \sum_{i=1}^{k} \max_{i \neq j} R_{ij} \, . \tag{8}$$

A smaller value of the Davis-Bouldin index, with zero being the lowest, indicates better separated clusters [22].

2.4 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a special type of neural networks which have become very popular for solving machine learning problems involving images [24] [25] [26]. The idea is that rather than representing images with handcrafted features, which can be both time consuming and difficult without extensive domain specific knowledge, the CNN learns to extract useful features by itself from labeled training data. While the architecture of different CNNs vary, they usually consist of three basic components - 2D convolutional kernels, non-linear activation functions and pooling operators.

The name giving component of a CNN is the 2D convolution between a kernel and an input image. The kernel is typically a 3x3 or 5x5 matrix, which is convolved with a much larger image to form linear combinations of adjacent pixel values. A mathematical description of the operation is given by

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{9}$$

where S is the output of the convolution, often referred to as a "feature map", K is the kernel and I is the input image.

Depending on the weights in the convolutional kernel, it may enhance certain structures in the input image such as corners, edges or circles. Usually, several kernels are used in parallel on the input image, producing many different feature maps as output. For color images, there are also three separate color channels of input for the kernels to operate on so that even more feature maps are generated. The feature maps themselves might in turn be convolved with another set of kernels, building up new feature maps of more complex structure. An important property of the convolutional kernels is that they are trainable. This means that the kernel weights are updated through training to extract features which are useful for solving the task of the CNN.

As most neural networks, the CNN has non-linear activation functions which perform some non-linear transformation of the input to the layer. A common choice for a CNN is the Rectified Linear Unit (ReLU), which sets all negative values of the input to zero, while all non-negative values are simply mapped to themselves

$$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0\,. \end{cases} \tag{10}$$

The third important component of a CNN is a pooling operator. A pooling operator takes as input a few nearby outputs from convolution and compresses them into a single summary statistic. For example, the max pooling operator divides a feature map into a grid with usually 2x2 or 3x3 elements in each cell. Only the maximum value in each cell is kept, thus reducing the size of the feature map considerably - a feature map of size 16x16 would after a 2x2 max

pooling be reduced to size 8x8. Aside from reducing the overall complexity of the neural network, pooling also adds the benefit of making the network more robust against image translations. Max pooling is the most commonly used pooling type, but there are other pooling operators such as average pooling, weighted average pooling and L^2 norm pooling.

A typical CNN begins with an input image, then a block which consists of a convolution, a non-linear activation and a pooling operator. The block structure is repeated a number of times. This part of the CNN is sometimes referred to as the feature extractor. At the end (the top) of the CNN sits a normal dense neural network, which takes as input the features extracted by the feature extractor and use these to classify the input image. When training the CNN, the weights of the dense network as well as the kernel weights are optimized to increase the classification performance [27].

2.4.1 Data augmentation

Data augmentation is a technique for increasing the number of training samples, without the need to collect more data [28] [29] [30]. Instead, the existing data is manipulated in different ways to create new samples which are similar to the original data, but not quite the same. It is important to balance the augmentation, so that new data is close enough to the original to be representative, but not so close as to cause overfitting of the machine learning system. In the context of image classification, data augmentations may be e.g. translations of the image, rotation, skewing, flipping, adding slight noise, dropping out pixels, cropping, color shifting and zooming, etc [31].

2.4.2 Feature extraction using pretrained convolutional networks

There exists a collection of publicly available CNNs with a rather deep and complex architecture, which have been trained on millions of natural images in the ImageNet [32] database to classify them into 1000 categories, ranging from cats and dogs to pencils and coffee cups. By removing the classifying part of these pretrained CNNs while keeping the feature extracting part with already well trained weights, the networks can be used to obtain useful general image features. These features can be used as input for another machine learning task, such as a different classifier. Depending on the application one might also choose to use the output from some intermediate layer of the feature extractor, to lock the pretrained weights or have them trainable for fine tuning. A benefit of using pretrained CNNs is that a lot of the computationally heavy training is already completed. Thus, it is possible to build powerful image processing applications even with limited computational resources.

A large variety of pretrained neural networks with different architectures, such as AlexNet, VGG, DenseNet, GoogLeNet and ResNet, have been proven successful at the task of classifying ImageNet images. An overview of some of these networks can be seen in table 3. For the task of feature extraction, pretrained networks of the ResNet family have been proven to be particularly useful [33] [34].

Network	Depth	Size (MB)	Parameters (10^6)	Input size
AlexNet	8	227	61.0	227-by-227
VGG16	16	515	138	224-by-224
VGG19	19	535	144	224-by-224
GoogleNet	22	27	7.0	224-by-224
DenseNet-201	201	77	20.0	224-by-224
ResNet18	18	44	11.7	224-by-224
ResNet50	50	96	25.6	224-by-224
ResNet101	101	167	44.6	224-by-224

Table 3: An overview of a selection of pretrained neural networks [33]. The table contains the network name, layer depth, size of stored weights, number of trainable parameters and required image input size.

2.4.3 Residual networks

Making neural networks deeper, i.e. increasing the number of layers stacked upon each other, is a common approach to increase performance and allow the network to succeed at more complex tasks. However, this is not entirely unproblematic. As deep networks become deeper, the backpropagated gradient of the loss function becomes increasingly smaller, a problem referred to as the vanishing gradient problem. The vanishing gradient has adverse effects on the training speed and convergence of the network. Solutions such as normalized weight initialization and intermediate normalization layers have been successful in mitigating this problem. Another problem which arises when the network becomes deeper is that of accuracy degradation. Training accuracy gets saturated and then degrades as more layers are added, contrary to what might be expected in terms of model complexity and overfitting. An architecture called residual networks (ResNets) has been successful in overcoming this problem [35]. What makes ResNets different from ordinary CNNs is the addition of shortcut connections, which allow the output from a convolutional layer to bypass a few layers and connect directly to the network at a later stage. An illustration of a shortcut connection is shown in figure 6.

Isolating a small part of an ordinary network $\mathbf{F(x)}$ which is an approximation of the mapping from the input, $\mathbf{x}$, to the output, $\mathbf{H(x)}$, gives

$$\mathbf{F}(\mathbf{x}) \approx \mathbf{H}(\mathbf{x}) \,, \tag{11}$$

adding a skip connection gives instead the mapping

$$\hat{\mathbf{F}}(\mathbf{x}) + \mathbf{x} \approx \mathbf{H}(\mathbf{x}) \,, \tag{12}$$

and equivalently:

$$\hat{\mathbf{F}}(\mathbf{x}) \approx \mathbf{H}(\mathbf{x}) - \mathbf{x} \,. \tag{13}$$

The last relation is what has given the ResNet family its name - the mapping of the residual $\mathbf{H(x)} - \mathbf{x}$. ResNets are constructed using these skip connection blocks. Shallower models, having fewer than 50 layers, use 2-layer blocks, whereas deeper models with a depth of 50 or more instead have 3-layer blocks.

The reason for this increase in block depth is to decrease training time.

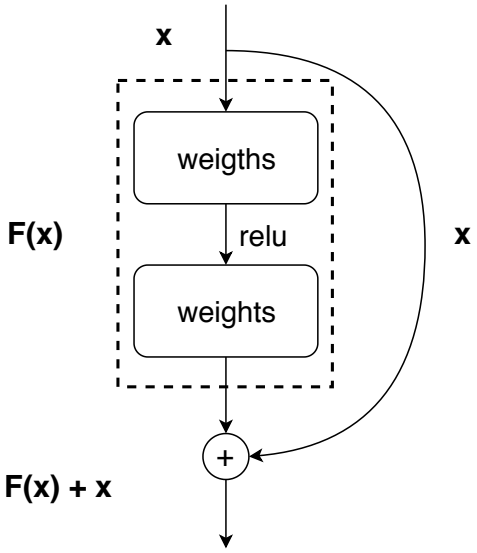

Figure 6: Illustration of the skip layer connection of ResNets.

2.5 ROI-Based Processing

In image analysis, identifying a region of interest (ROI) is partly done to avoid the need for processing a full image. By defining ROI boundaries all the pixels outside the area will be ignored. The boundaries are fixed or updated gradually. The pixels in the ROI can be filtered with a threshold for classification or manipulated in any other desired way. This will speed up the computation and enables enhancing important information only visible in certain regions. ROI-based processing is a common approach for medical imaging, where the boundaries help focusing on an object, e.g. the movement of an organ [36] [37]. Other interesting areas range from fingerprint segmentation [38], to developing pedestrian detection systems on vehicles [39].

2.6 Optical flow

Optical flow is the resulting 2D vector field from comparing two consecutive frames and tracking motion of objects, as can be seen in figure 7. The vectors represent the displacement of an object from its position in the previous frame. Assumptions regarding optical flow are that objects retain their pixel intensities during movements, and that a patch of neighbouring pixels has a comparative movement [40].

Figure 7: Optical flow visualized with flow vectors representing moving people [41].

2.6.1 Gunnar Farnebäck algorithm (Dense optical flow)

The Gunnar Farnebäck algorithm [42] estimates the displacement between two frames to discover motion, and it is a dense technique since it calculates the flow vectors for all individual pixels (compared to the Lucas–Kanade approach of using a sparse feature set [43]). The neighbourhood of a pixel is approximated by the following polynomial

$$f(\mathbf{x}) \sim \mathbf{x}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{x} + c\,, \tag{14}$$

where $\mathbf{A}$ is a symmetric matrix, $\mathbf{b}$ a vector and c a scalar. A weighted least squares fit estimates the coefficients. An ideal translation of a polynomial (i.e. a movement of a neighborhood of pixels, e.g. from a ball flying in the air) is calculated by comparing two polynomials at the same coordinate in two subsequent frames. The difference is the displacement. This is done by constructing the signal f_1 from the first frame

$$f_1(\mathbf{x}) = \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + \mathbf{b}_1^T \mathbf{x} + c_1\,, \tag{15}$$

and f_2 from the subsequent frame

$$\begin{aligned} f_2(\mathbf{x}) &= f_1(\mathbf{x} - \mathbf{d}) = (\mathbf{x} - \mathbf{d})^T \mathbf{A}_1 (\mathbf{x} - \mathbf{d}) + \mathbf{b}_1^T (\mathbf{x} - \mathbf{d}) + c_1 \\ &= \mathbf{x}^T \mathbf{A}_1 \mathbf{x} + (\mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d})^T \mathbf{x} + \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1 \\ &= \mathbf{x}^T \mathbf{A}_2 \mathbf{x} + \mathbf{b}_2^T \mathbf{x} + c_2\,, \end{aligned} \tag{16}$$

where $\mathbf{d}$ is the global displacement. From the last two lines in equation 16 the coefficients are determined

$$\begin{aligned} \mathbf{A}_2 &= \mathbf{A}_1, \\ \mathbf{b}_2 &= \mathbf{b}_1 - 2\mathbf{A}_1 \mathbf{d}, \\ c_2 &= \mathbf{d}^T \mathbf{A}_1 \mathbf{d} - \mathbf{b}_1^T \mathbf{d} + c_1\,, \end{aligned} \tag{17}$$

and from this, $\mathbf{d}$ is calculated as

$$\mathbf{d} = -\frac{1}{2} \mathbf{A}_1^{-1} (\mathbf{b}_2 - \mathbf{b}_1)\,. \tag{18}$$

Two polynomials at the same coordinate in two frames might differ more than the displacement. This is an issue especially for larger movements. A priori

knowledge can be incorporated to handle these movements to reduce the error. The equations presented here have assumptions of an ideal scenario to give a brief introduction to the concept. To understand the algorithm in more detail readers are referred to [42].

2.7 Software

TensorFlow [44] is an open source machine learning platform. It uses the concept of computational graphs to allow users to build complex pipelines for data with distribution over several computational nodes. It is directed both at machine learning researchers to help advance the field of machine learning, as well as developers to deploy machine learning powered applications. TensorFlow is developed by Google and was released as open source in 2015.

Keras [45] is an API for constructing, training and evaluating neural networks, which is built to run with TensorFlow [44], CNTK [46] or Theano [47] as backend. It provides a library for deep learning that supports many types of neural networks including CNNs. It also provides support for data augmentation, piecewise loading of large datasets, pretrained neural networks and more. It was originally developed by François Chollet.

OpenCV [48] is an open source library for computer vision with many algorithms and functions related to image analysis and computer vision. Apart from basic functions to read and write images and video in a program, it supports applications such as facial recognition, 3D model extraction, object tracking and augmented reality.

2.7.1 Multicore processors

Optimizing the computational time is necessary to maximize the gain of using machine learning. This can be done by improving the algorithms or improving the hardware [49]. A multicore processor enables parallelism which can decrease the computational time. Multiple processing units can operate simultaneously and therefore process a larger amount of data compared to a single core processor. Another advantage for battery driven appliances is the reduction of power consumption with two cores instead of one. Using several processing units does not automatically reduce the computational time. It is necessary to construct the machine learning algorithm in a multicore compatible way, to prevent starvation or idle cores [50].

3 Data

The data used in this thesis consists of 9 video recordings, further described in table 4. The videos were recorded at three hospitals. All the videos depict percutaneous heart procedures. This thesis does not dive into any details about the procedure, only using the basic information that can be retrieved from the available visual material (operating table, staff, screens, blanket, blood, instruments).

Video	Hospital	Length (hh:mm:ss)	Number of frames
1	1	00:54:19	81497
2	1	02:58:52	268313
3	1	00:59:15	88876
4	1	02:17:04	205612
5	1	02:55:13	262833
6	2	00:21:48	32720
7	2	01:29:58	134957
8	2	00:58:12	87309
9	3	01:19:54	119871

Table 4: Information about the videos used in this thesis.

3.1 Video sources

The videos had views from four time synchronized sources, as shown in figure 8. Video sources 1 and 2 were cameras capturing different overview angles of the procedure, corresponding to "Overview video" in figure 1. The angles changed between each video, but the content was similar. Source 1 captured a wide angle of the room. Source 2 was a close up of the operating table. Source 3 displayed fluoroscopy video, corresponding to "X-ray video" in figure 1. Source 4 monitored patient parameters, but was not used in this thesis. The environment, staff and colours of the equipment were varying since the recordings came from different hospitals.

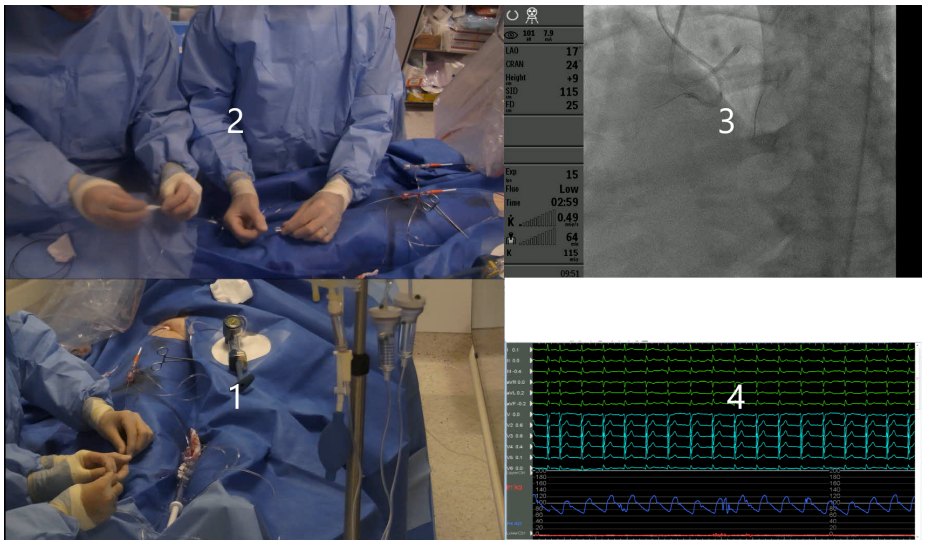

Figure 8: A sample image from video 3, with four time synchronized video sources showing a percutaneous heart procedure.

3.2 Cross-validation datasets for tool 3

To develop tool 3 with supervised training of a CNN, the data was manually annotated. Extracted frames were divided into seven folds for a 7-fold cross-validation scheme. Table 5 shows the relationship between the video numbers and the constructed datasets. Due to a major movement of the camera, video 7 was divided into two different folds as there was such a large difference in the frames. Video 1, 2 and 5 did not contain anything to annotate, and were therefore excluded from this set.

Video	View	Validation in dataset
3	2	1
4	2	2
6	2	3
7a	2	4
7b	2	5
8	2	6
9	1	7

Table 5: Validation video and corresponding dataset used for 7-fold cross-validation. The views correspond to the numbers in figure 8.

4 Method

To speed up the computation of the algorithms described below, an NVIDIA GeForce GTX 1070 Ti graphics card was used to enable parallelization. The code was written in Python using JupyterLab and the Python libraries Keras [45] (with Tensorflow [44]), Scikit-learn [51] and OpenCV [48].

4.1 Tool 1: Unsupervised clustering

The first method was a cluster based technique without ground truth. The idea was to find patterns in the frames, and structure these into chapters. No a priori assumptions were made, making the algorithm generalizable for many types of medical procedures. All videos where clustered individually. The output of tool 1 was a vector with numbers where the index represented a frame and the number represents its cluster label. Each cluster can be seen as a chapter of the video.

4.1.1 Frame preprocessing

OpenCV was used to extract each frame from the video. Preprocessing the input is a recommended procedure when using pretrained models [52]. In the case of ResNet50 pretrained on ImageNet, this meant resizing each frame to 224x224 pixels (from the original 1920x1080 resolution), changing the order of the color channels from RGB to BGR and subtracting the mean of the ImageNet color channels: $B = 103.939$, $G = 116.779$, $R = 123.68$, from each pixel. The rearranging of color channels and subtraction of mean color was facilitated with the built-in `preprocess_input()` function associated with Keras ResNet50 [53] [54].

4.1.2 Feature extraction with ResNet50

After the preprocessing, ResNet50 was used to extract features containing information about shapes and structures in each frame. Instead of representing a frame by all its 1920x1080 pixels, it was represented by these features. A complete ResNet50 resulted in 2048 features, decreasing the amount of data to 1‰ to use for clustering.

A deep network produces specific features with detailed information of common patterns in the ImageNet dataset. To generalize the features, a shallow ResNet50 with 30 layers using only a fifth of the trainable parameters was also computed, producing 1024 features per frame.

4.1.3 k-means++ clustering

The k-means++ algorithm was used to organize the frames of a video into clusters based on the similarity of the features, with the intention that frames with similar features should end up in the same cluster. Several clusterings of each video were calculated, changing the number of clusters $k = 3, 5, 7, 9, 11$, as well as the two different feature representations. In total, this means that for each video, 10 different clusterings where calculated. For each clustering, the

Silhouette, Calinski-Harabaz and Davies-Bouldin scores were calculated to aid in the estimation of the best value for k.

4.1.4 Median filter

The clusters are meant to represent specific phases of the procedure and a good clustering means that frames from the same time period in the video should appear in the same cluster. The output from the clustering was sometimes noisy with peaks appearing at random places. These peaks are frames that have been paired together in a cluster with other frames from a different time period of the video. A median filter was used to remove these peaks. With a window size of 1001, the 500 frames before and after a specific frame on the time axis were observed, which corresponds to 20 seconds before and after. The new cluster label for the frame was the median of the cluster labels of these 1000 neighbouring frames.

4.1.5 Sort clusters

On the timeline the clusters should resemble an ascending stair starting with cluster 1, but since the cluster labels were not sorted this was not the case, which can be seen in the centre plot in figure 13. This was solved by renaming each cluster depending on its position on the timeline. To find the positions, the mean of the cluster frame indices were compared. In the centre plot in figure 13 the cluster order is 5, 1, 4, 3, 2. The mean index of cluster 5 is the lowest mean and therefore cluster 5 is renamed to cluster 1. The mean index of cluster 1 is the second lowest mean and therefore renamed to cluster 2, and so on. The sorted cluster labels are the output of tool 1, which can be seen in the lower plot in figure 13.

4.2 Tool 2: X-ray detector

Tool 2 was a feature based technique with the purpose to detect frames from the fluoroscope where the X-ray source was active. Iterating through the video, a vector of zeros and ones was built, where a zero indicated no detection in the corresponding frame, and a one indicated that the X-ray was detected.

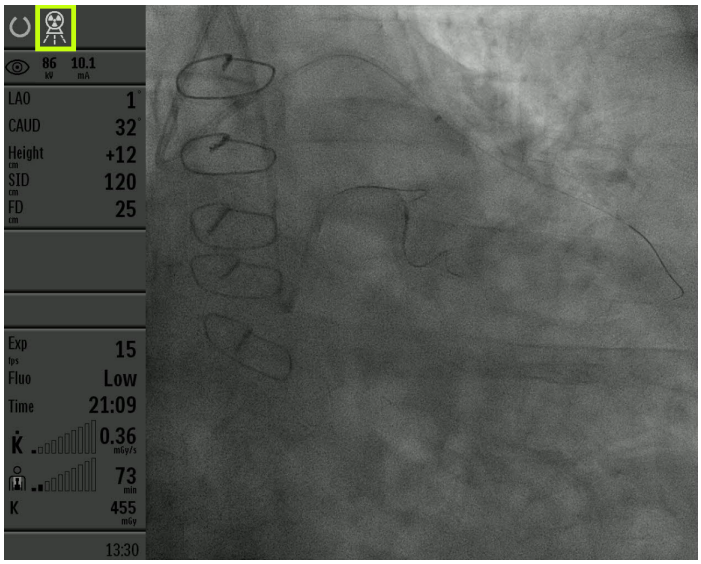

Figure 9: The image shows an X-ray frame with a bounding box within which the intensity is counted to capture if the symbol is present. In this case it is.

Individual frames from the video were extracted using OpenCV. In all the videos, the layout of the X-ray frames were similar. Of particular interest was a symbol located in the top left corner, which is present when the X-ray is active. See figure 9 for reference.

Having manually located the X-ray symbol, a ROI bounding box was created around it. The average intensity of the pixels within the box was calculated. A threshold was used to determine if the symbol was present. Studying the average pixel values with (207 average value) and without (183 average value) the X-ray symbol present, it was determined that a threshold at 195 was adequate for making the detection.

4.3 Tool 3: Supervised instrument detector

Tool 3 was a feature based technique with the purpose to identify when an object of clinical interest is present. The output was a vector of zeros and ones where a one indicated that the object had been detected in the corresponding frame.

4.3.1 The Optical Coherence Tomograph - OCT

The first step towards training a supervised classifier on finding objects was choosing an object to detect. Considerations such as our limited medical expertise, availability of training material and resolution of video led to the choice of an instrument used for optical coherence tomography (OCT). OCT is used during percutaneous heart procedures for imaging inside the blood vessels, when the guidance provided by external X-ray is not enough [55] [56] [57]. The importance of OCT during percutaneous heart procedures, in addition to it being present in 6 of the 9 videos and straightforward to recognize, made it a good

candidate for devising a supervised instrument detector. In figure 10 a sample frame with the OCT present is shown.

Figure 10: In this frame the OCT is present.

4.3.2 Video annotation

The 6 videos in which the OCT was present were manually annotated as either being with OCT, without OCT or ambivalent. Ambivalent frames where e.g. the OCT is temporarily occluded or just entering/leaving the frame were excluded from training. This was motivated by wanting only unambiguous training samples for the classifier to improve its performance. Ambivalent frames were few, and therefore the potential limited capability of classifying them were not considered to be of much importance for the purposes of this thesis.

4.3.3 7-fold cross-validation

The initial dataset of annotated OCT frames consisted of 49619 positive and 605997 negative examples from the 6 applicable videos. The dataset was divided into 7 folds, for a 7-fold cross-validation scheme as illustrated in figure 2. The folds were constructed so that each of the 6 videos corresponded to one fold, with the exception of video 7 which was divided into two separate folds due to a major shift in camera angle halfway through the video. In table 5 in the Data section the relationships between video and fold is listed.

To avoid redundancy of similar neighbouring frames and to balance the amount of samples in each fold and the ratio of negative and positive samples, the folds were evenly subsampled so that 1700 positive and 1700 negative frames were kept from each fold. The number 1700 was selected as the smallest fold (fold 6 coming from video 8) contained only 1742 positive samples.

4.3.4 Training a CNN

A CNN with the architecture shown in figure 11 was selected as an OCT detector. The network was trained with the settings shown in table 6. Following the 7-fold cross-validation scheme, 7 models were trained with a different fold left out for validation for each model.

Because of limitations in memory, all training images could not be loaded into the program at once. Instead, the Keras function `flow_from_directory()` to flow images on demand from a directory was utilized. In conjunction with this the training images were randomly augmented to increase the diversity. The augmentation settings used are listed in table 7. Note that augmentation was used only when training the classifier, and not during validation.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 32)      9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 43264)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2768960
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 2,797,665
Trainable params: 2,797,665
Non-trainable params: 0
_________________________________________________________________
```

Figure 11: The CNN architecture used in the OCT classifier.

Setting	Value
Epochs	15
Batch size	170
Learning rate	0.001
Optimizer	Adam
Loss	Binary cross-entropy
Intermediate activations	ReLU
Final activation	Sigmoid

Table 6: Settings used for training the OCT classifier in tool 3.

Augmentation setting	Value
Rescale	1/255
Rotation range	20°
Width shift range	0.2
Height shift range	0.2
Horizontal flip	True
Zoom range	0.2
Shear range	0.2
Brightness range	0.7-1.3

Table 7: Settings for Keras data augmentation when training the OCT classifier.

4.3.5 Validation and predictions

To evaluate the performance of the CNN, the validation accuracy was calculated for each model, as well as the mean validation accuracy across all models. Additionally, a ROC curve and AUC score was calculated for each model, to better understand the model and allow a more flexible threshold selection. See figure 19 and table 11 in the Results section.

Apart from evaluating how well the CNN classified the frames in the constructed datasets, it was also of interest to see how it performed on an entire video with some ambivalent cases where the OCT is partially occluded or almost in the frame (i.e. when it is entering or exiting the scene).

The 6 videos where the OCT was present were processed frame-by-frame by the classifier. Each video was classified by a model which had not been trained on frames from that video before. Video 7 (fold 4 and 5) was a special case, where the CNN had been trained on the first and second half of the video respectively prior to its predictions. The predictions of the CNN were compared with the ground truth. A decision boundary for labeling a frame as with or without OCT was manually selected from the ROC-curve in figure 19 for each video. Low false positive rate was prioritized, since false positives will confuse the timeline. If the OCT is present in a sequence of 15 minutes, the tool should guide the clinician to where that sequence is on the timeline, but detecting all OCT frames in that sequence is not critical. Therefore, the true positive rate does not have to be maximized.

4.4 Tool 4: Camera motion detector

The aim of tool 4 was to detect camera movements. When people enter the room there is often (in these videos) an adjustment of the camera position. When the procedure starts, a zooming to get a better view of the procedure is not unusual. An example of a camera adjustment is given in figure 12.

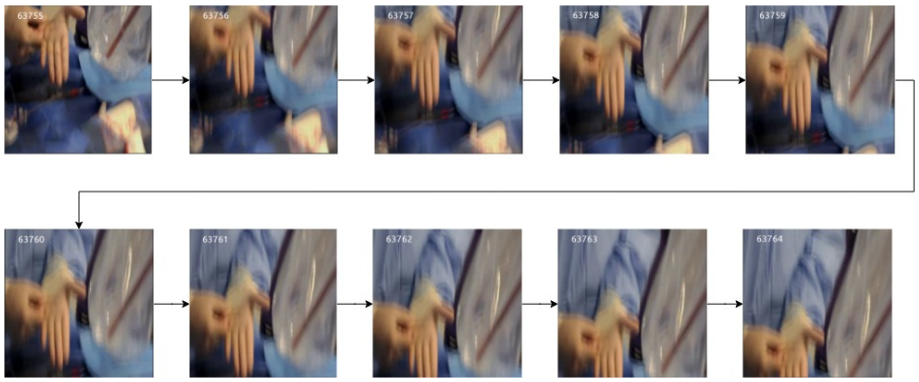

Figure 12: A series of frames showing global movement of the camera. All pixels of the frame are moving, as opposed to a local motion where only a part of the frame (e.g. the hand) would be moving.

4.4.1 Farnebäck optical flow

Frames were extracted with OpenCV. In order to estimate the movement between two subsequent frames, the Gunnar Farnebäck algorithm for optical flow was used. The algorithm calculates the magnitude and direction of flow for each pixel.

4.4.2 Distinguishing between local and global motion

It is necessary to distinguish between a global motion, such as when the camera is moved or zoomed in, from local movements such as a person moving in an otherwise stationary scene. This problem becomes particularly difficult in zoomed in scenes, where for example a moving hand (local motion) takes up a large portion of the frame and might therefore easily be confused with a global motion.

The solution was to calculate how many pixels in a frame were moving. A pixel was classified as moving if the flow magnitude was above 1 (heuristically chosen). If the total number of moving pixels was over a certain threshold (chosen to be 80% from the ROC-curve in figure 23), the frame-to-frame motion was defined as global, otherwise local. Only the global motions were of interest since the aim was to detect camera movements.

4.4.3 Removing single peaks

Occasionally the videos froze, which caused errors. If the duration of a freeze is e.g. ten frames, the difference between the last frame in the freeze and the subsequent frame will be like comparing a movement of ten frames at once, which will look like a fast global motion and appear as a single peak in the plot. Since the freeze is the cause and not an actual camera movement, it is undesired. With a median filter this category of single peaks were suppressed, see center plot in figure 22.

4.4.4 Validating the classifier

Annotating the camera adjustments and zoom in the video made it possible to evaluate how often a global motion was discovered by the classifier. The threshold to separate global and local motion was varied to calculate a ROC-curve and AUC score.

4.5 Automatic summary

While the four tools of this thesis can be used individually, it was also of interest to investigate how they can be combined to create a representation of a medical video with a lot of information delivered to the user in a compact format. For this task, two kinds of summaries were created: a combined timeline and a short video summary.

4.5.1 Combined timeline

The combined timeline was constructed by plotting the output from tool 1, 2, 3 and 4 together into the same graph, illustrated in figure 24 and figure 25. With a common time basis, the timeline shows the partitioning into different clusters and when the X-ray is active, the OCT is present and there is global motion in the video. The staircase format was chosen to represent the partitioning into clusters, whereas points of different colors were used to represent X-ray, OCT and global motion.

4.5.2 Short video summary

To decrease the length of the original full video, only a few frames were chosen to be presented in a summary. A rate for how many frames to skip was calculated for each cluster generated by tool 1. The formula is seen beneath in equation 19. From each cluster, the same number of frames were included. A longer cluster will generate a higher rate, and more frames will be skipped in that cluster in the summary.

$$\text{cluster } i \text{ rate} = \frac{\text{cluster } i \text{ length} \cdot \text{numbers of clusters}}{\text{total number of frames} \cdot \alpha} \tag{19}$$

Once the base set of frames had been selected, the X-ray, OCT and global motion detections were added. The tools were weighted, to choose how much of each tool should be included, e.g. an X-ray weight of 0.5 meant that every 2:nd frame with active X-ray should be included in the summary.

Finally, information about time, chapter, X-ray, OCT and global motion status was added to each frame. Transitions from one chapter to another were marked out by temporarily making the chapter number larger 25 frames before and after the transition.

5 Results

5.1 Tool 1: Unsupervised clustering

The results of using tool 1 on video 7 with $k = 5$ and $k = 7$ are shown in figure 13 and 14. For both figures, the general features corresponding to the shallow version of ResNet50 were used.

Figure 15 and 16 show the same clusterings as above, with a frame representing each cluster.

Table 8 shows the cluster metric scores for each video and value of k when using the specific features of a complete ResNet50. Table 9 shows the same metrics, using the general features from the shallow network.

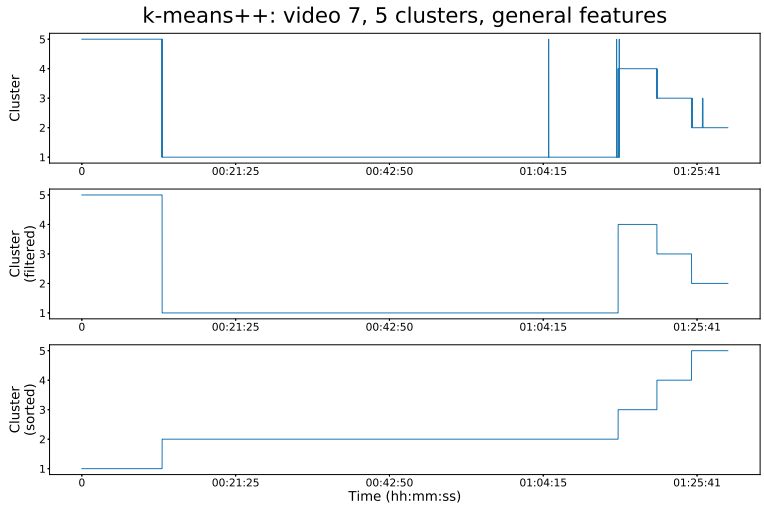

Figure 13: The upper plot is the output from k-means++ clustering on general features of video 7 with 5 clusters. In the center plot the output has been filtered to remove noise. The lower plot is the final output from tool 1, where the clusters have been sorted.

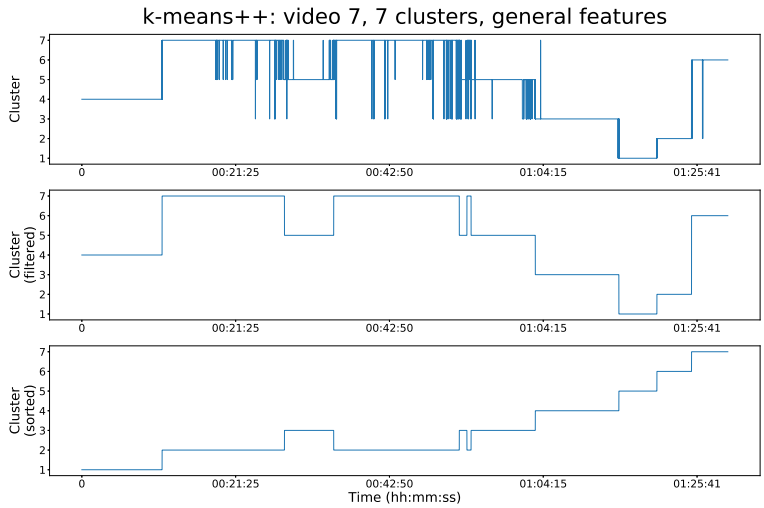

Figure 14: The upper plot is the output from k-means++ clustering on general features of video 7 with 7 clusters. In the center plot the output has been filtered to remove noise. The lower plot is the final output from tool 1, where the clusters have been sorted.

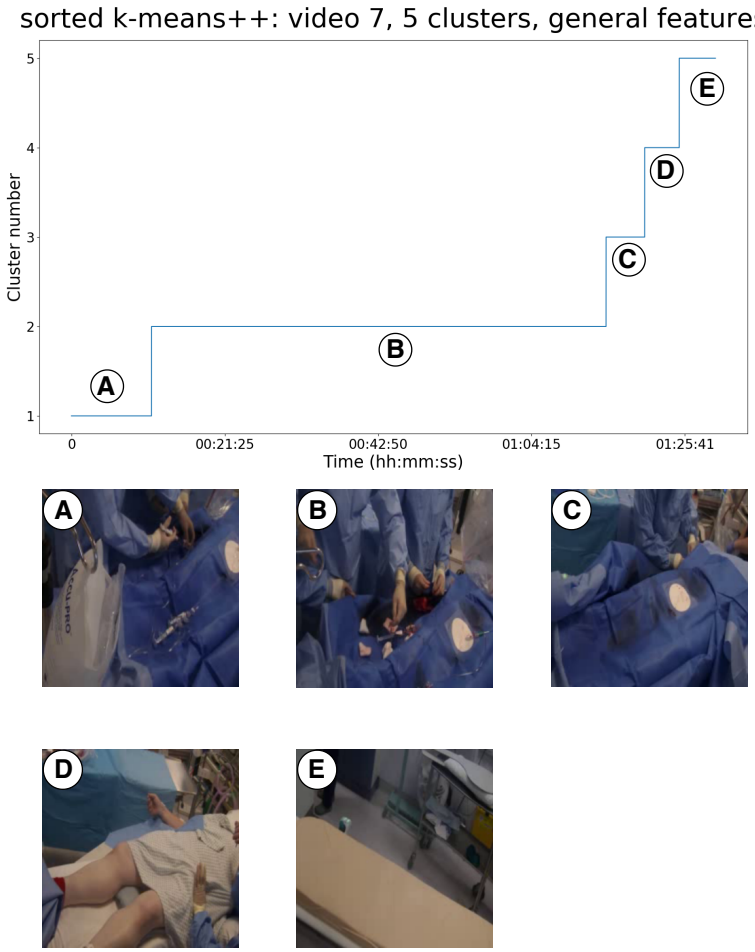

Figure 15: The output of tool 1 on video 7 with 5 clusters, visualized with frames.

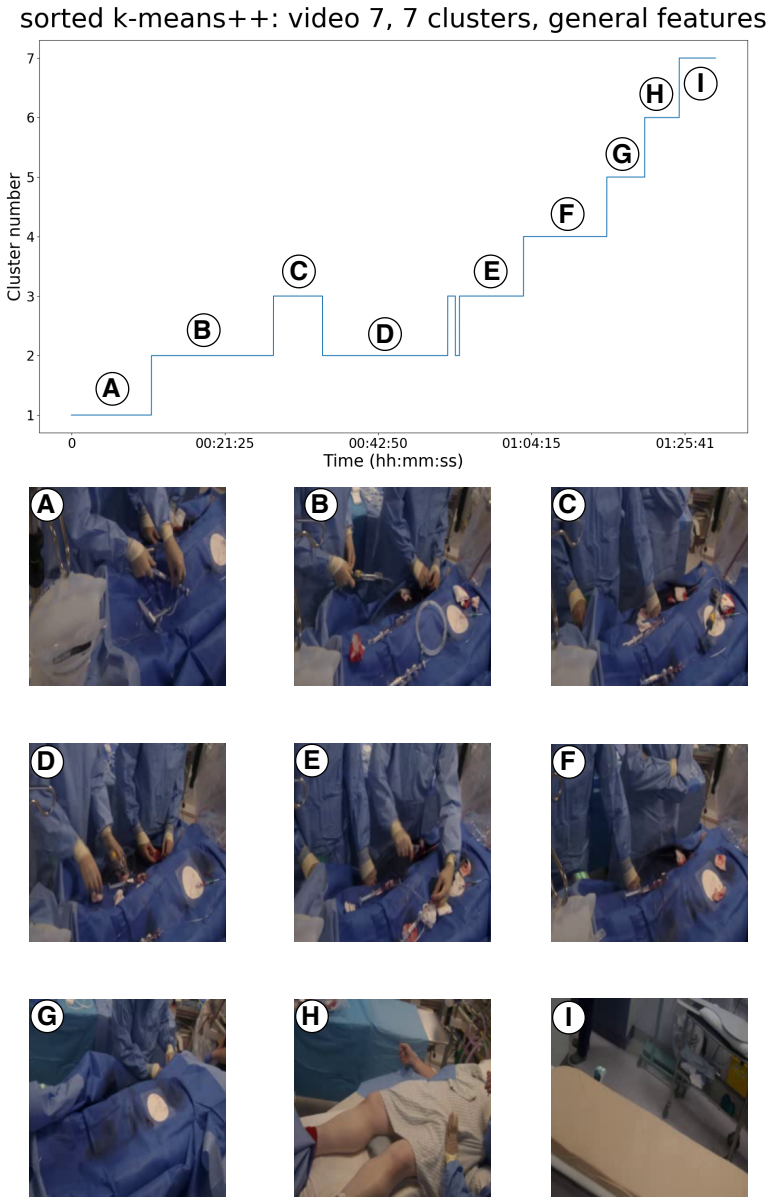

Figure 16: The output of tool 1 on video 7 with 7 clusters, visualized with frames. Each letter does not correspond with a unique cluster since some clusters reoccur (B/D and C/E).

Specific	k=3	k=5	k=7	k=9	k=11
Video	S CH DB	S CH DB	S CH DB	S CH DB	S CH DB
1	0.17 13162 2.24	0.13 10281 2.21	0.14 8679 2.29	0.14 7421 2.30	0.13 6533 2.37
2	0.12 29656 2.73	0.08 21295 2.80	0.08 17484 2.99	0.09 14926 2.83	0.09 12953 2.80
3	0.07 6946 2.98	0.09 5942 2.76	0.10 5255 2.51	0.10 4620 2.61	0.11 4187 2.48
4	0.13 30003 2.27	0.08 22110 2.58	0.08 17367 2.76	0.08 14355 2.93	0.08 12428 2.91
5	0.27 63917 1.73	0.31 58888 1.46	0.21 49076 2.02	0.22 41396 1.80	0.19 37375 1.88
6	0.15 5775 1.81	0.19 4702 2.26	0.18 3756 2.31	0.19 3283 2.13	0.19 2904 2.00
7	0.16 22512 1.83	0.12 17134 2.35	0.13 14622 2.11	0.13 13294 2.16	0.15 12117 2.12
8	0.13 12442 2.12	0.16 9599 2.24	0.15 8086 2.27	0.14 6949 2.39	0.12 6182 2.52
9	0.10 12083 2.63	0.09 9104 2.51	0.08 7370 2.74	0.08 6354 2.94	0.07 5590 2.99

Table 8: Silhouette (S), Calinski-Harabaz (CH) and Davis-Bouldin (DB) score for 3-11 clusters on specific features. In each cell, the metrics are presented in the order S, CH, DB.

General	k=3	k=5	k=7	k=9	k=11
Video	S CH DB	S CH DB	S CH DB	S CH DB	S CH DB
1	0.25 13776 1.63	0.11 10114 2.23	0.11 8283 2.48	0.12 7097 2.33	0.14 6465 2.30
2	0.15 44234 2.24	0.13 32633 2.27	0.12 26713 2.53	0.13 22411 2.48	0.12 19590 2.54
3	0.11 8311 2.81	0.11 7111 2.41	0.10 6282 2.37	0.10 5586 2.41	0.10 4995 2.46
4	0.46 120005 0.78	0.12 82100 2.03	0.09 59045 2.59	0.10 46967 2.79	0.10 39514 2.71
5	0.44 134887 1.16	0.38 131333 1.22	0.37 116712 1.10	0.36 98155 1.25	0.26 87415 1.52
6	0.20 5763 2.35	0.16 4364 2.35	0.17 3714 2.16	0.18 3295 2.00	0.18 2972 2.07
7	0.26 55395 1.65	0.26 42469 1.33	0.16 33596 1.88	0.17 28167 1.94	0.18 25457 1.86
8	0.18 19539 1.96	0.19 16265 1.96	0.18 13742 1.94	0.15 11603 2.02	0.15 10187 2.19
9	0.25 15109 1.81	0.09 11962 2.42	0.08 9605 2.61	0.08 8064 2.40	0.08 7039 2.53

Table 9: Silhouette (S), Calinski-Harabaz (CH) and Davis-Bouldin (DB) score for 3-11 clusters on general features. In each cell, the metrics are presented in the order S, CH, DB.

5.2 Tool 2: X-ray detector

In figure 17 and 18 the result of using tool 2 on video 1 and video 4 is shown. Table 10 shows the ratio of active and inactive X-ray frames for each of the 9 videos.

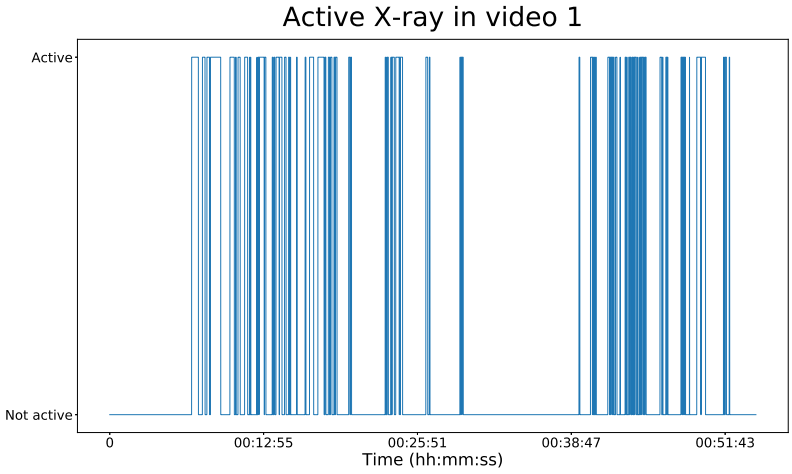

Figure 17: Visualization of when the X-ray is active in video 1.

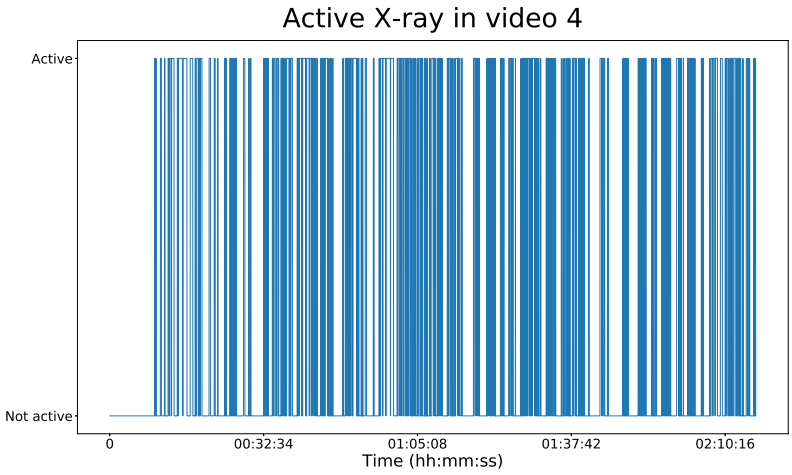

Figure 18: Visualization of when the X-ray is active in video 4.

Video	Active X-ray [%]
1	19
2	54
3	25
4	35
5	42
6	16
7	18
8	28
9	34

Table 10: Overview of how often (percentage of full video) the X-ray was activated.

5.3 Tool 3: Supervised instrument detector

The result from tool 3 is presented in figure 19 with a ROC-curve, and as an overview in table 11.

Figures 20 and 21 show the OCT classifications on video 1 and 4. The blue lines correspond to the annotation. "Leave out" means that the instrument is present, but not visible enough to be used for training of the network. The upper plot shows the classifiers predictions. A threshold was used to determine whether or not a prediction was enough for a positive classification. The thresholds were different for each video and were extracted from the ROC-curves in figure 19 by looking at the highest ratio between true positive rate and false positive rate.

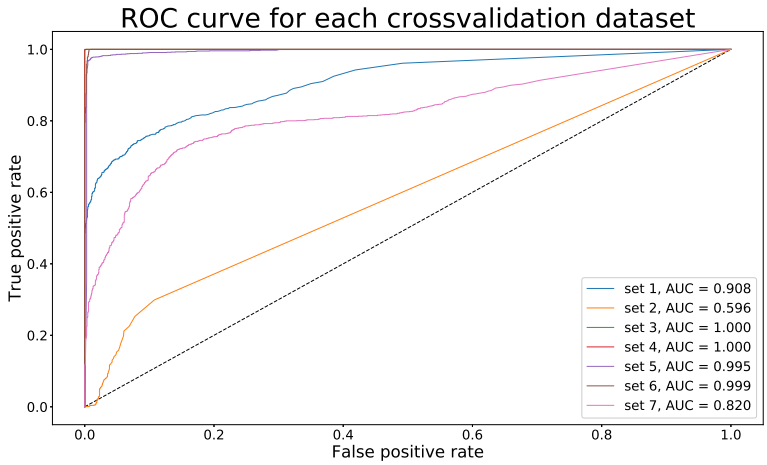

Figure 19: ROC-curve for all seven cross-validation trainings on the OCT.

Cross-validation set	Accuracy	AUC
1	82.0	0.908
2	50.1	0.596
3	99.3	1.000
4	98.5	1.000
5	98.0	0.995
6	89.2	0.999
7	89.3	0.820
Average	**86.6**	**0.936**

Table 11: Accuracy for each of the cross-validation datasets when training the OCT network.

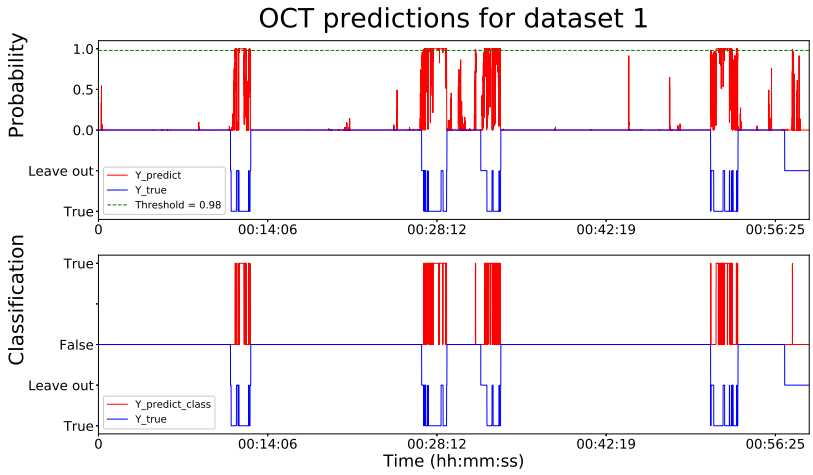

Figure 20: The upper plot shows the prediction by the classifier on dataset 1. The lower plot has a threshold at 0.98 for a positive classification. Both plots are compared to the annotation of the OCT, which is represented by the blue lines.

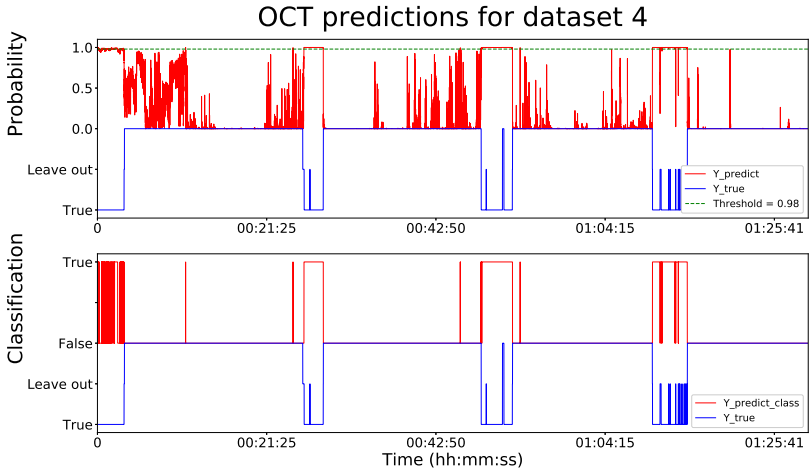

Figure 21: The upper plot shows the prediction by the classifier on dataset 4. The lower plot has a threshold at 0.98 for a positive classification. Both plots are compared to the annotation of the OCT, which is represented by the blue lines.

5.4 Tool 4: Camera motion detector

The result of using tool 4 on video 2 is shown in figure 22. A ROC-curve where all 9 videos have been processed by tool 4 is shown in figure 23.

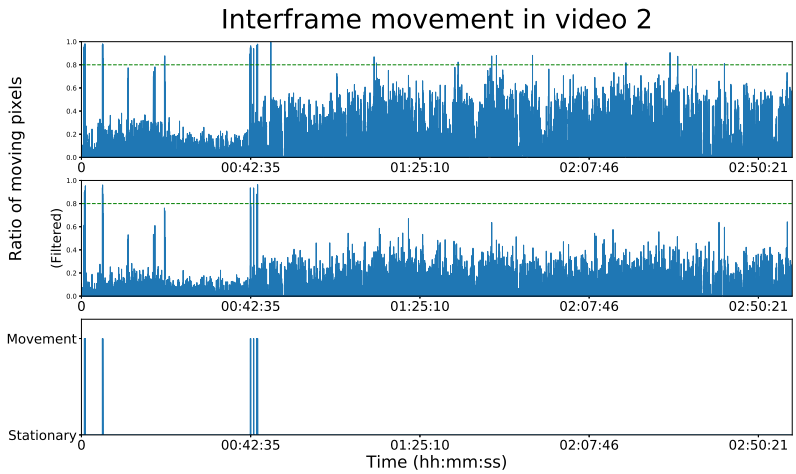

Figure 22: Optical flow signal from video 2 before filtering, after filtering and after thresholding at 0.8 (the dashed green line). If 80% of the pixels in a frame are moving, it is classified in the third plot as a camera movement.

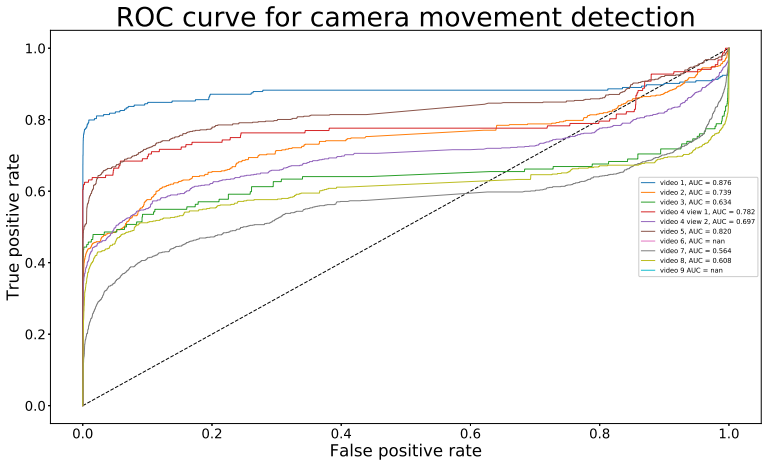

Figure 23: ROC curve for detection of camera movement and zoom. Video 6 and 9 had no movements or zooms, therefore the ROC-curve is not applicable for these videos.

5.5 Automatic summary

The combined timelines of video 6 and 7, clustered on general features, are shown in figure 24 and 25. Figure 26 shows a video summary frame from video 7.

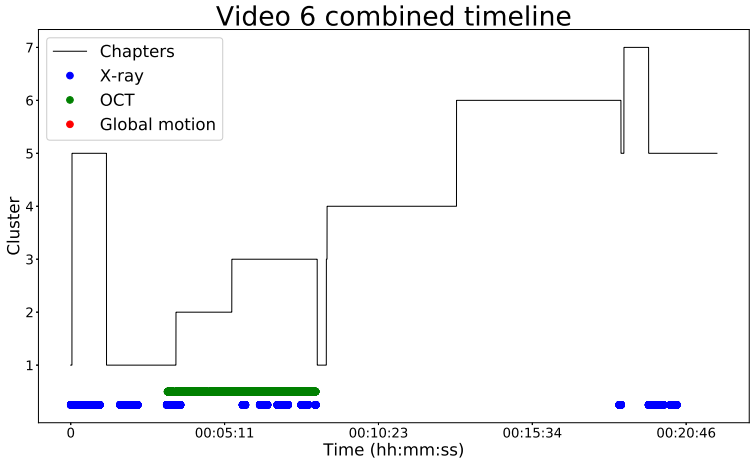

Figure 24: The four tools extracted from video 6 combined. Tool 1 was used with 7 clusters in the construction of this timeline.

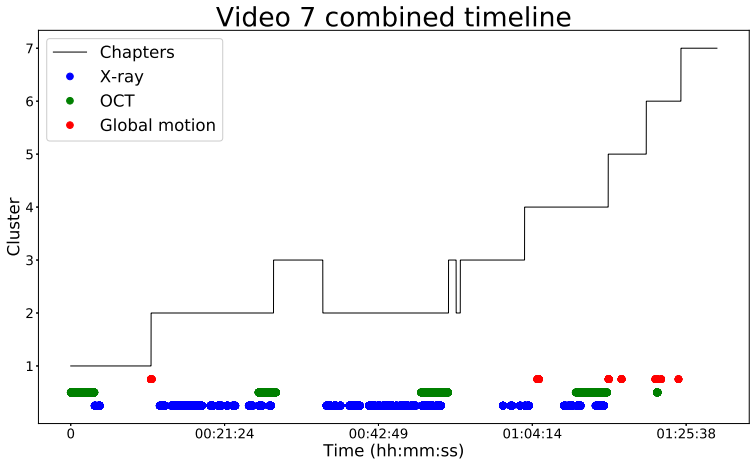

Figure 25: The four tools extracted from video 7 combined. Tool 1 was used with 7 clusters in the construction of this timeline.

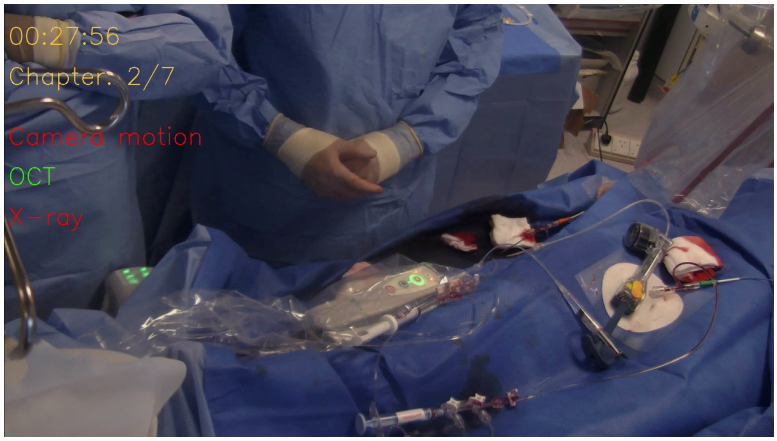

Figure 26: A frame from the summary of video 7, with information about time, chapter, OCT, X-ray and camera motion. Red colour means not active/present, and green means active/present.

6 Discussion

To find suitable tools to show the promise of an intelligent toolbox for video editing, we started by discussing with Medical Imaging Technologies, and came to the conclusion that an unsupervised tool without any a priori information should be included. The reason was that there was no annotated data, and an unsupervised tool could be generalized to many medical procedures, not only the percutaneous heart procedures in the dataset. This resulted in tool 1.

From researching the field of automatic video summarization we found three categories that were suitable for the video material: feature, cluster and shot selection based techniques. Feature based techniques were represented by finding the X-ray (tool 2) and the OCT (tool 3). Finding camera movements (tool 4) was based on shot selection. Cluster based techniques were covered by tool 1. Tool 2-4 were meant to show the potential of tailoring a toolbox based on the procedure and the interest of the clinician. The OCT was of interest for percutaneous heart procedures, but could be substituted with another instrument. The reason was to show the effectiveness of neural networks, though it required annotated frames. The active X-ray and the camera movements added extra information.

Knowing and understanding a clinicians need is essential when constructing a toolbox. We got in contact with Kiet Tran [58], a heart surgeon at Skånes Universitetssjukhus in Lund, Sweden, who gave his thoughts on the subject. He told us that identifying an instrument is of great interest since his surgeries include stopping the heart, where a tong is an identicator of when the critical part begins. Identifying the entry and removal of the tong in the video would therefore be good reference points. This could be done by annotating and using a neural network like tool 3. The team around K. Tran writes down when the patient enters the room and other logistical information. Connecting these timestamps to the video would do the same job as tool 1. The question is if our tool can replace the need for writing down information and connecting it to the video. Regardless, K. Tran thought the information to be of a more logistical character, and not something that is prioritized to see for a surgeon, compared to the tong.

In retroperspective we should have had Mr. Tran's thoughts before we started the thesis, but it took time to find a contact. We started from scratch with an idea from Medical Imaging Technologies, and researched the field and found tools relevant to previous work. Now this thesis and K. Tran's opinions will be a good start for the next master thesis to continue on this topic.

6.1 Data

The data of this thesis was nine medical videos of percutaneous heart procedures from three different hospitals. Writing robust algorithms requires a lot of data, and it is difficult to know if nine videos are enough. The amount of frames is almost 1.3 million, but since they come from videos several frames are nearly identical. The performance of tool 3 might give a hint on the need for more data. The average accuracy, as seen in table 5 in the Results section, was 86.6%

- by "only" using six videos for training. The result indicates that it is possible to create a decent classifier from a few videos. What lowers the average accuracy is the 50% performance on cross-validation set 2, which corresponds to video 4. The brightness in video 4 differs from the other videos. We tried to solve this with data augmentation, but it was unsuccessful. To develop the tools further it would be a good idea to acquire more data with different settings.

6.2 Tool 1: Unsupervised clustering

Extracting features with a pretrained CNN and then using k-means++ to cluster the frames based on deep features turned out well. Although the clustering algorithm has no knowledge or concept of the temporal dependence between frames, it still groups the frames more or less in order so that the frames belonging to the same period in time are assigned to the same group, as is shown in figure 13 and figure 16. This indicates that the visual content of the videos changes enough over time so as to make this approach viable. Examples of how the visual contents change between clusters are shown in figure 15 and figure 16.

Tool 1 is ideal to use on videos with pre and post sequences of a procedure, since large changes in the environment are reflected in the clustering. A video only recording when the procedure is running and is zoomed in is not ideal since not even we know how to divide it up into chapters without knowledge of the procedure. It is hard to expect more from an algorithm. Our aim has been to at least identify major changes in the room. The algorithm distinguishes between an empty room and a room with staff. It notices when the patient leaves the operating table. Big instruments will generate new clusters. Combining tool 1 with an editing user interface would make it possible to throw away a cluster displaying an irrelevant sequence like an empty room.

6.2.1 Benefit of the ResNet50

It would be possible to do a k-means++ on all the pixels instead of using extracted features from ResNet50, but it requires more memory, increases the computational time and comparison is limited to pixels instead of complex visual structures. Clustering on all the pixels would mean having 2073600 elements (1920x1080) instead of 2048 (or 1024 with a shallow ResNet50).

The features of ResNet50 are specialized on the patterns of the ImageNet dataset. To generalize the features, a ResNet50 with a reduced number of layers was also used. It was difficult to decide how many layers to use. Using a fewer number of layers should result in more basic features, such as edges and corners, but there is a limit where the features become too general. An example of the difference in clustering on general and specific features can be noticed by comparing figure 13 with 27 in the Appendix.

6.2.2 Median filter: Time structure vs visual similarity

By applying the median filter the temporal coherence of clusters is enhanced, but the visual similarities upon which the clustering is based is ignored. This might lead to a loss of important information, such as when a fast transient

event is occurring in the original video. Such events might be of interest, as things which do not occur often could be just as important as the ordinary activity of the video. An extension to the clustering-based approach could be to include those frames which were forced to change cluster by the filtering.

6.2.3 Sorting clusters

If frames both in the beginning and end of a video are similar, they tend to end up in the same cluster. The mean frame of that cluster is then in the middle of the video, which causes problems when sorting the clusters. This occurs in figure 24.

6.2.4 Finding k in k-means

The selection of k in the k-means++ algorithm is an important decision to make in tool 1. If k is large, the algorithm might be forced to split "natural" chapters apart, whereas too few clusters have the effect of joining chapters together when it would be more natural to separate them. Examples of the issue can be seen in figure 28 and 29 in the Appendix, where the clustering was made on 3 and 9 chapters.

Manual decision of the best number of chapters can be made by looking at an entire video or visually examining the output of tool 1 for different values of k. However, these manual approaches are not well suited for an automated system.

Using the Silhouette, Calinski-Harabaz and Davis-Bouldin metrics for cluster evaluation was an attempt to objectively decide an optimal number of chapters. While it provided some guidance in e.g. video 5, where both the Silhouette and Davis-Bouldin score indicate that 5 clusters is the best choice when using more specific features, the outcome is mostly inconclusive. For many of the videos, the metrics vary only slightly when k is changing, thus conveying no guidance at all. Another common behaviour is that the metrics indicate that the lowest amount of clusters is the best choice, which defeats the purpose of dividing the video into partitions in the first place. See table 8 and table 9 for reference.

From a video summary point of view, the number of frames in each chapter should perhaps be taken into account, as it is a reasonable wish that a video of 2 hours length should be divided into more chapters than a short clip of 20 minutes. One might also try to use different features from those used in this thesis, in case they prove more suitable for partitioning the videos.

6.3 Tool 2: X-ray detector

Detecting the X-ray is a clue to understanding what part of a video is interesting to see. Several heart procedures, and especially those in this thesis, use catheters together with X-ray to examine the heart [59]. Clusters far away from an active X-ray flag on the timeline is an indicator of inactivity.

The method of finding the active X-ray may seem hard coded. We manually identified an area in the videos and counted the pixels within the area. After

thresholding, the output was either active or not active, as can be seen in figure 17 and 18. If the layout of the frame changes or the pixel intensities were somehow different, the method would break. A future version of this tool might be more dynamic and e.g. ask the user to mark out the location of the symbol, or use automatic detection. Another way could be to obtain a signal from the fluoroscope itself rather than relying on the video signal, although at the cost of more wiring and the need for interpreting that signal instead.

6.3.1 Percentage of active X-ray

Table 10 shows how often the X-ray is active in a video. This could be an additional feature for the clinician, to indicate how much the X-ray is being used.

An issue is that the percentage is dependent on when the recording begins and ends. If the camera is running around-the-clock, the value does not convey much information. It is useful only if the routine is to start and stop the camera at the beginning and the end of a medical procedure, or if the beginning and end of a procedure is marked out.

6.4 Tool 3: Supervised instrument detector

Table 11 and figure 19 indicate that the OCT classifier performed well on the majority of the validation datasets, with the exception of dataset 2 which is illustrated in figure 30 in the Appendix. Keep in mind that the dataset numbers are not the same as the video numbers (explained in the Data section). Dataset 2 corresponded to video 4. Compared to the other videos, video 4 was a lot brighter in color, which is a likely cause for the bad performance. Some augmentation of the training images was attempted to overcome this problem, but without success. Here is potential for future work, e.g. by obtaining more diverse data to train a classifier on, or by improving the augmentation. Permutation of the color channels or working with gray scale images might solve the problem.

6.4.1 Flexibility of tool 3

The tool showed how useful neural networks can be to find objects in a video. We chose to annotate the instrument for OCT as it was present in many of the available videos and is important to percutaneous heart procedures. The instrument could be substituted to suit videos of other types of procedures. A drawback of this tool is the need for annotated data for training. We had to spend a few days to annotate data.

While the OCT is a comparatively large object and not very hard to detect for a human, other interesting objects might be harder to annotate and train a classifier on. Detection of smaller objects such as scalpels and needles might be hampered by e.g. obstruction by the hand holding the instrument. Image resolution could also affect the performance, and it might be necessary to use the full 1920x1080 pixels rather than the 224x224 pixels that were used in this

thesis. Increasing resolution comes at a cost of memory requirements and longer training times.

6.5 Tool 4: Camera motion detector

Considering the result in figure 23, the method for detecting global camera motion was better than randomly guessing, though it by no means perfectly matched the annotated movements. Video 6 and 9 had no camera movements, hence the low ratio of moving pixels in figure 31 in the Appendix, which our classifier correctly rejected as "no movement".

In part, this might be connected to the difficulty in consistently annotating where there was movement, where it begun and where it ended. If a prolonged movement over 50 frames in fact consists of many shorter movements, with just a few stationary frames in between, it is tempting to annotate the entire sequence as one movement. The detector would likely see it as many small movements however, as it is only comparing two adjacent frames at a time. A similar problem arises when the camera "sways" for a long time in the same position, which occurred a few times. This should ideally not be picked up by the detector.

The ROC curve in figure 23 has a peculiar shape in the top right corner, where it goes below the dotted line which represents random guessing. The likely explanation is that there are camera movement frames with a ratio of moving pixel both above and below the stationary frames in figure 22. As the threshold has passed below the first group of the camera movement frames, the true positive rate in the ROC curve stabilizes. As the threshold decreases, the false positive rate goes up, until there are more false positives than true positives. At some point, the threshold reaches the camera movement frames with low ratio of moving pixels, and the true positive rate increase again. The difficulties in annotating movement consistently might also be a cause for this effect.

6.6 Automatic summary

The automatic summary is a way of combining all previous tools and presenting their outputs in a compact format. The combined timeline, as shown in figure 24 and 25 is an example of how this could be done in a static way, providing an overview of the video with points of interest marked out. This way of presenting information could be useful in a video editing software, where it is common to have a timeline of the video for navigation, cutting, merging clips, adding audio etc.

The other proposed way of presenting the video is making a video summary, almost like a trailer, with the information about time, chapter, X-ray, OCT and global motion printed on each frame. A challenge in the development of an automated summary, is to define what constitutes a "good summary". In the best of worlds, one or a few videos annotated by a trained clinician could serve as a gold standard. The summary could be evaluated by comparing the output of the system to the gold standard set by the trained clinician. Another solution could be to have a panel of intended end users expressing their opinion

on different summary proposals. This is beyond the scope of this thesis, and more suited towards a designer. The purpose of our summary was to act as a proof of concept of the use of the tools.

7 Conclusions

The aim of this master thesis was to create an intelligent toolbox for editing medical video and to use the toolbox to generate a video summary. Four tools were to be created, inspired by existing techniques for automatic video summarization.

Tool 1 was a clusterer, which was difficult to evaluate as no annotated data was available. The independence from ground truth is also one of the appeals of this tool, as it could be adapted for many situations without extensive data preparation. Tool 2 was a feature based X-ray detector, which was implemented by counting pixels in a region of interest. Tool 3 was a supervised classifier, using a convolutional neural network to detect when a medical instrument was present. An optical coherence tomograph was selected as the instrument of interest. Overall the classifier performed well, but when lighting conditions where too different from the training data the performance was no better than random guessing. Tool 4 was a global camera motion detector, based on optical flow. Camera translations were detected, but slow zoom passed by undetected.

The importance of the tools were discussed with a surgeon who showed interest in identifying instruments, corresponding to tool 3. Tool 1 could be interesting for logistical planning. A timeline showing the information from all tools simultaneously and a short video summary was created to achieve the secondary goal of this thesis.

7.1 Future work

We have a few suggestions for the further development of the toolbox. Our ideas are presented in the following sections.

7.1.1 Investigate what tools are in demand

Retrieving more input from clinicians or intended users will build a good knowledge of what is important in a video summary. Using the tools in practice will not only require relevant tools, but also a focus on design to make the information accessible and easy to handle.

7.1.2 Improvements to tool 1

The implementation of tool 1 in this thesis uses k-means++ clustering as a basis for automatic partitioning into chapters. An assumption is made that frames which belong together visually also belong together in time, but apart from that the clusterer does not take any consideration to the temporal order of the frames. An interesting development could be a clustering scheme which makes use of the temporal order. Testing other features could also be of interest. ResNet50 was chosen as it is considered good for image feature extraction in general, but it is not specifically trained on images in a medical setting. Another problem which could be addressed is finding a way to determine how many chapters a video should be divided into.

7.1.3 Improvements to tool 3

Training the algorithm on more data can make it more robust. An alternative is to expand the use of data augmentation to cover more scenarios, or to use a different CNN architecture.

7.1.4 Adapt tool 3 for another instrument

The principle of using neural networks with annotated data could be used with other medical instruments. The possibility to track any instrument is of interest when tailoring the video summary for specific medical procedures. Increasing resolution might be necessary to track needles or other small equipment.

7.1.5 Improvements to tool 4

A future development of tool 4 could be to replace the threshold on the optical flow signal with a sliding window and a trainable classifier such as logistic regression. Using a sliding window the state of neighbouring frames would be taken into account as well, not just the value of the single frame.

7.1.6 Video summary with reinforcement learning

In the article *Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity-Representativeness Reward* [60] a reinforcement learning scheme for creating summaries is suggested which tries to make the frames of a summary as diverse as possible, while also keeping the summary representative. Implementing the reinforcement learning scheme for medical videos could be an interesting alternative to the toolbox and the generation of a video summary proposed in this master thesis.

References

[1] Raghupathi W and Raghupathi V. *Big data analytics in healthcare: Promise and potential.* Health information science and systems, 2014.

[2] Wilhelmsson, P. Engineer with ten years of expertise from the surveillance industry. Personal interview, 2019-02-19.

[3] Medical Imaging Technologies
https://www.medicalimagingtechnologies.com/ [Accessed 2019-02-08]

[4] Medical Imaging Technologies press-release 2019-01-07
https://www.arabhealthonline.com/en/media/press-releases/why-hospitals-need-audio-visual-integration.html [Accessed 2019-02-08]

[5] Plexus Surgical Video Production
https://plexus.tv/ [Accessed 2019-02-08]

[6] SurgiCast *Medical Video Editing*
https://www.surgicast.io/services/video-editing [Accessed 2019-02-08]

[7] Incathlab
https://www.incathlab.com/en/home [Accessed 2019-02-08]

[8] Ajmal M, Husnain Ashraf M, Shakir M, Abbas Y and Ali Shah F. *Video Summarization: Techniques and Classification.* Proceedings of the International Conference of Computer Vision and Graphics, pp. 1-13, 2012.

[9] Louridas P and Ebert C. *Machine Learning.* IEEE Software, vol. 33, pp. 110-115, 2016.

[10] Mohri M, Rostamizadeh A and Talwalkar A. *Foundations of Machine Learning, second edition.* Series: Adaptive computation and machine learning, chap, 1.5/4.5, 2012.

[11] Fawcett T. *An introduction to ROC analysis.* Pattern Recognition Letters, v.27, n.8, pp. 861-874, 2006.

[12] Ben Ayed A, Ben Halima M and Alimi A M. *Survey on clustering methods: Towards fuzzy clustering for big data.* 6th International Conference of Soft Computing and Pattern Recognition, pp. 331-336, 2014

[13] Scikit Learn - 2.3 Clustering. https://scikit-learn.org/stable/modules/clustering.html [Accessed 2019-05-14]

[14] Ester M, Kriegel H. P, Sander J and Xu X. *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise.* Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 226–231, 1996

[15] L.R Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition.* Proceedings of the IEEE, vol. 77, pp. 257-286, 1989

[16] Ghassempour S, Girosi F and Maeder A. *Clustering Multivariate Time Series Using Hidden Markov Models.* International Journal of Environmental Research and Public Health, vol 11(3), pp. 2741–2763, 2014

[17] Kubat M. *An Introduction to Machine Learning.* Springer Cham, pp. 273-295, 2017.

[18] Reddy Edla D, Tripathi D, Kuppili V and Cheruku R. *Survey on Clustering Techniques.* Second International Conference on Inventive Communication and Computational Technologies, pp. 696-703, 2018.

[19] Arthur D and Vassilvitskii S. *k-means++: The Advantages of Careful Seeding.* Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms, pp. 1027-1035, 2007.

[20] Rousseeuw P. *Silhouettes: A graphical aid to the interpretation and validation of cluster analysis.* Journal of Computational and Applied Mathematics, vol. 20, pp 53-65, 1987.

[21] Calinski T and Harabasz J. *A Dendrite Method for Cluster Analysis.* Communications and Statistics - Simulation and Computation, 1974.

[22] Davies D and Bouldin D. *A Cluster Separation Measure.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol PAMI-1 no 2, pp. 224-227, 1979.

[23] Scikit Learn - 2.3.9. Clustering performance evaluation `https://scikit-learn.org/stable/modules/clustering.html` [Accessed 2019-04-09]

[24] Zhuangzi L, Xiaobin Z, Lei W and Peiyu G. *Image Classification Using Convolutional Neural Networks and Kernel Extreme Learning Machines.* 25th International Conference on Image Processing, 2018.

[25] Török P and Harangi B. *Digital Image Analysis with Fully Connected Convolutional Neural Network to Facilitate Hysteroscopic Fibroid Resection.* Gynecologic and Obstetric Investigation, pp. 615-619, 2018.

[26] Tianqiang P, Yongwei Z and Shengcai K. *Image retrieval based on convolutional neural network and kernel-based supervised hashing.* 8th International Congress on Image and Signal Processing, 2015.

[27] Goodfellow I, Bengio Y and Courville A. *Deep Learning.* MIT Press, ch.9, 2016.

[28] Chaoyun Z, Pan Z, Chenghua L and Lijun L. *A Convolutional Neural Network for Leaves Recognition Using Data Augmentation.* International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2015.

[29] Alani A, Cosma G, Taherkhani A and McGinnity M, *Hand gesture recognition using an adapted convolutional neural network with data augmentation.* 4th International Conference on Information Management, 2018.

[30] Qiucheng D, Aiguo W, Na D, Wei F and Shaohua W. *A Convolution Neural Network for Parts Recognition Using Data Augmentation.* 13th World Congress on Intelligent Control and Automation, 2018.

[31] Keras - Image Preprocessing, https://keras.io/preprocessing/image/ [Accessed 2019-04-26]

[32] ImageNet http://www.image-net.org/ [Accessed 2019-02-08]

[33] MathWorks - Pretrained Convolutional Neural Networks https://se.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html [Accessed 2019-02-08]

[34] Kornblith S, Shlens J and Quoc V L. *Do Better ImageNet Models Transfer Better?* Google Brain, 2018.

[35] Kaiming H, Xiangyu Z, Shaoqing R and Jian S. *Deep Residual Learning for Image Recognition.* IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[36] Region of interest (ROI) https://www.techopedia.com/definition/339/region-of-interest-roi [Accessed 2019-04-23]

[37] Brinkmann R. *The Art and Science of Digital Compositing.* Morgan Kaufmann, p. 184, 1999.

[38] Stojanovic B, Neskovic A, Popovic Z and Lukic V. *ANN based fingerprint image ROI segmentation.* TELFOR, pp. 505-508, 2014.

[39] Mesmakhosroshahi M, Loghman M and Joohee Kim. *Feature-based ROI generation for stereo-based pedestrian detection.* ICASSP, pp. 1727-1731, 2017.

[40] OpenCV: Optical Flow https://docs.opencv.org/trunk/d7/d8b/tutorial_py_lucas_kanade.html [Accessed 2019-04-26]

[41] An Introduction to the NVIDIA Optical Flow SDK https://devblogs.nvidia.com/an-introduction-to-the-nvidia-optical-flow-sdk/ [Accessed 2019-04-26]

[42] Farnebäck G. *Two-frame motion estimation based on polynomial expansion.* Proceedings of the 13th Scandinavian conference on Image analysis, 2003.

[43] Lucas B and Kanade T. *An iterative image registration technique with an application to stereo vision.* Proceedings of Imaging Understanding Workshop, pp. 121-130, 1981.

[44] TensorFlow, https://www.tensorflow.org/ [Accessed 2019-04-26]

[45] Keras, https://keras.io/ [Accessed 2019-04-26]

[46] CNTK, https://docs.microsoft.com/en-us/cognitive-toolkit/ [Accessed 2019-04-26]

[47] Theano, http://deeplearning.net/software/theano/ [Accessed 2019-04-26]

[48] OpenCV, https://opencv.org/ [Accessed 2019-04-26]

[49] Serpa MS, Krause AM, Cruz EHM, Navaux POA, Pasin M and Felber P. *Optimizing machine learning algorithms on multi-core and many-core architectures using thread and data mapping.* EMPDP, pp. 329-333, 2018.

[50] Sethi A. *Multicore processor technology- advantages and challenges.* International Journal of Research in Engineering and Technology. v.4, n.9, pp. 87-89, 2015.

[51] Scikit-learn, https://scikit-learn.org/stable/ [Accessed 2019-05-08

[52] Keras feature extraction example, https://keras.io/applications/#classify-imagenet-classes-with-resnet50 [Accessed 2019-04-30]

[53] Keras ResNet50 preprocessing, https://github.com/keras-team/keras-applications/blob/master/keras_applications/resnet50.py [Accessed 2019-04-30]

[54] Keras preprocess input source code, https://github.com/keras-team/keras-applications/blob/master/keras_applications/imagenet_utils.py [Accessed 2019-04-30]

[55] Terashima M, Kaneda H and Suzuki T. *The Role of Optical Coherence Tomography in Coronary Intervention.* The Korean Journal of Internal Medicine 27, pp. 1-12, 2012.

[56] Sharma S, Rijal J and Dahal K. *Optical coherence tomography guidance in percutaneous coronary intervention: a meta-analysis of randomized controlled trials.* Cardiovascular Intervention and Therapeutics 34, pp. 113-121, 2019.

[57] Longobardo L, Mattesini A, Valente S and Di Mario C. *OCT-Guided Percutaneous Coronary Intervention In Bifurcation Lesions.* Interventional Cardiology Review, vol. 14, n.1, pp. 5-9, 2019.

[58] Tran K. Heart surgeon and researcher in vascular surgery at Skånes Universitetssjukhus. Study visit and personal interview, 2019-04-24.

[59] Cardiac Catheterization https://www.heart.org/en/health-topics/heart-attack/diagnosing-a-heart-attack/cardiac-catheterization [Accessed 2019-04-09]

[60] Zhou K, Qiao Y, Xiang T. *Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward.* Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

8 Appendix

Tool 1

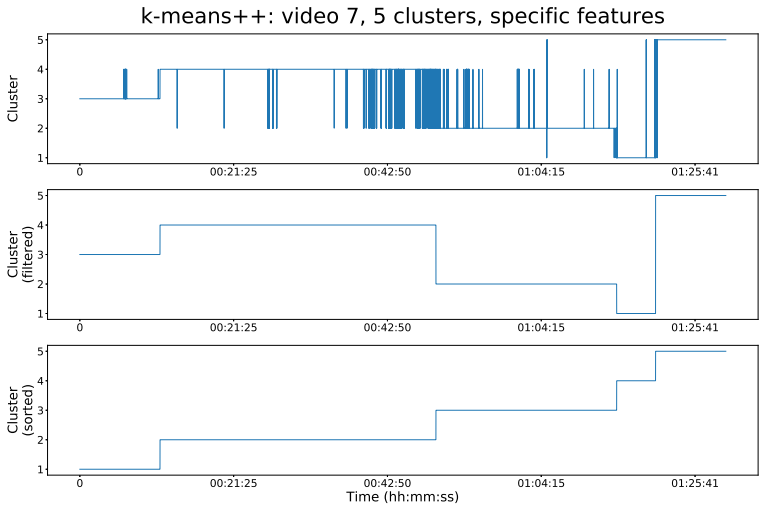

Figure 27: The upper plot is the output from k-means++ clustering on specific features of video 7 with 5 clusters. In the center plot the output has been filtered to remove noise. The lower plot is the final output from tool 1, where the clusters have been sorted.

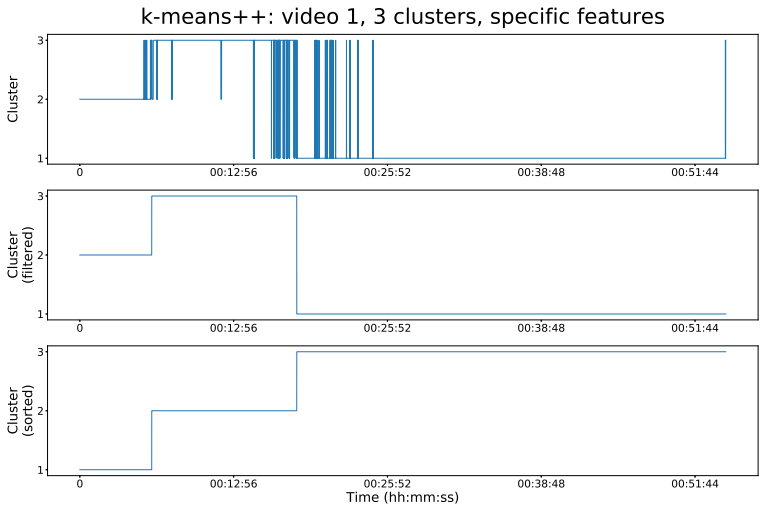

Figure 28: The upper plot is the output from k-means++ clustering on specific features of video 1 with 3 clusters. In the center plot the output has been filtered to remove noise. The lower plot is the final output from tool 1, where the clusters have been sorted.

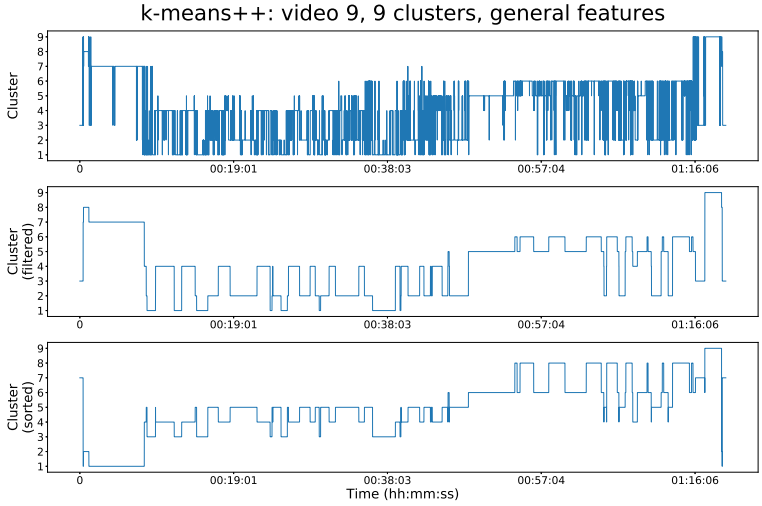

Figure 29: The upper plot is the output from k-means++ clustering on general features of video 9 with 9 clusters. In the center plot the output has been filtered to remove noise. The lower plot is the final output from tool 1, where the clusters have been sorted.

Tool 3

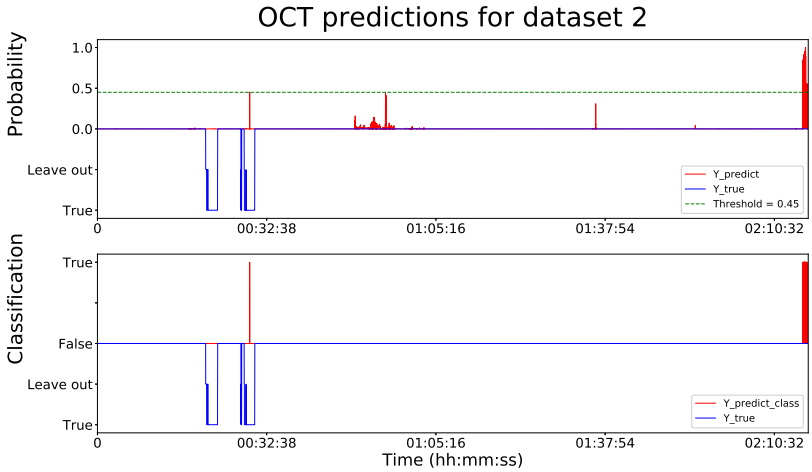

Figure 30: The upper plot shows the prediction by the classifier on dataset 2. The lower plot has a threshold at 0.45 for a positive classification. Both plots are compared to the annotation of the OCT, which is represented by the blue lines.

Tool 4

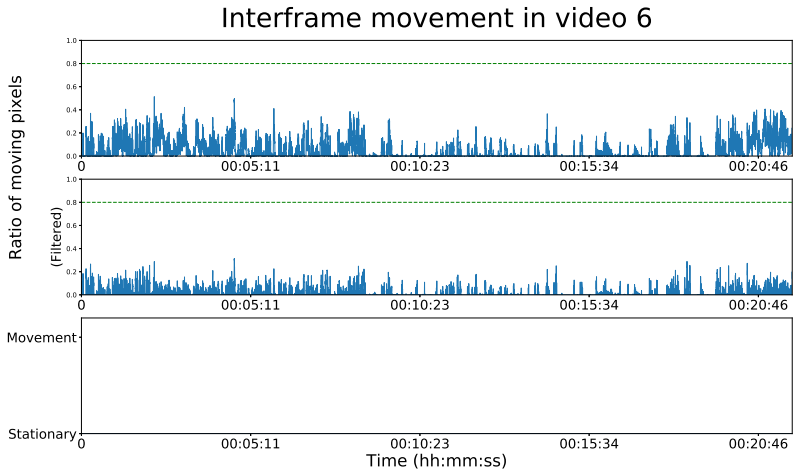

Figure 31: Optical flow signal from video 6, which had no camera movements.

Master's Theses in Mathematical Sciences 2019:E25
ISSN 1404-6342

LUTFMA-3383-2019

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/