Chapter 4

Implementation

In this chapter the implementation of the novel video fingerprinting system is discussed in detail, starting with frame fingerprinting and finishing off with video fingerprinting. Frame fingerprinting is divided into three subsections: algorithm, database and searching. The algorithm section explains how the hashes are calculated for a frame, using the SIFT algorithm. The database section explains the database structure and how hashes are saved. The searching section explains how an unknown frame's hashes are compared to the database to find a match in the minimal amount of time. Video fingerprinting consists of KFD and frame fingerprinting and is discussed at the end.

4.1 Frame Fingerprinting

In the previous chapter it was explained how the idea for the new video fingerprinting technique came into being and why the techniques that were discussed are important. In this section the new frame fingerprinting technique will be discussed and the incorporation of the existing techniques explained in detail.

The main focus was on developing an algorithm that can process and detect frame matches as fast as possible while still maintaining robustness and database efficiency. After researching the current video fingerprinting systems, it was evident that the focus of these systems were mainly on robustness, because of the environments the systems are used in. This means that the fingerprinting and detection times of the current existing techniques are quite high. For this application the assumption was made that the level of distortion on a television broadcast isn't very high, thus allowing the technique to be a little less robust to distortions than other existing techniques, but fast enough to run in real-time.

4.1.1 Algorithm

To start the frame fingerprinting process, a frame is sampled from the video, then it is normalised by re-sampling it to a set resolution of 320×240 and converting the resized image to gray-scale. By normalizing the frame, processing is simplified and lessened, because all the frames are the same size and uses the same colour space.

Key points for the frame are then calculated using the SIFT algorithm. The default parameters are used in the SIFT algorithm, except for the start octave variable, which was given a value of 2. This effectively causes the frame to be down-sampled twice before key points are calculated. This means that only the key points with the largest magnitudes in each frame are calculated. The key points with the largest magnitudes are used in the fingerprinting process, because they are more robust and repeatable than smaller key points and the processing time is greatly reduced if only the largest key points have to be calculated while the smaller, insignificant key points are skipped.

To create the set of hashes (fingerprint) of the frame, the key points are paired with other key points in the frame if they are within a certain distance from each other. The maximum distance is equal to the magnitude value of the primary key point and the minimum distance is a tenth of the maximum distance. This means that larger key points will be paired with more key points than smaller ones.

To ensure that larger key points are paired with each other before the smaller key points,

a novel tier system is used. The system works by dividing the set of key points into tiers, the 10 largest key points being the first tier, the next 10, the second tier, and so on. A primary tier and secondary tier is then selected and every key point in the primary tier is tested against every key point in the secondary tier to determine if they form a valid key point pair to create a hash. If the maximum number of hashes per frame is reached (value discussed in Subsection 5.2.2), the key point pairing ends, otherwise new primary and secondary tiers are selected to find more key point pairs to create hashes. The following piece of code shows how the primary and secondary tiers are selected (both start at the first tier(tier 0)):

```
'Resulting sequence of tiers (primary-secondary) : 0-0, 0-1, 1-1, 0-2, 1-2, 2-2,
' 0-3, 1-3, 2-3, 3-3, 0-4 ...
If primary_tier = secondary_tier Then
    primary_tier = 0
    secondary_tier += 1
Else
    primary_tier += 1
End If
```

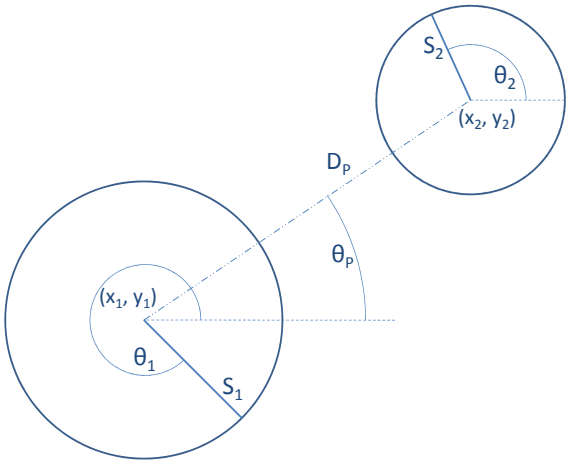

Figure 4.1: Diagram of two matched key points.

Figure 4.1 shows an example of two key points that are matched. S is the magnitude, θ the direction and (x, y) the coordinates of the key point. The following equations have been designed specifically for this application and they are used to calculate the values needed to create a hash from a key point pair:

- Magnitude ratio:
$$H_1 = \frac{S_2}{S_1} \tag{4.1}$$
- Relative direction:
$$H_2 = \theta_2 - \theta_1 \tag{4.2}$$
- Relative distance to secondary point:
$$H_3 = \frac{D_P - 0.1 \times S_1}{0.9 \times S_1} \tag{4.3}$$

  where $D_P = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. The relative distance is calculated within the range of 0.1 to 1 of the primary point's scale, because the points are not paired if their distance is below 0.1 of the primary point's scale.
- Relative direction to secondary point:
$$H_4 = \theta_P - \theta_1 \tag{4.4}$$

  where $\theta_P = \tan^{-1}(\frac{y_2 - y_1}{x_2 - x_1})$ if $x_2 - x_1 \neq 0$
  if $x_2 - x_1 = 0$ and $y_2 - y_1 > 0$, then $\theta_P = \frac{3}{2}\pi$
  if $x_2 - x_1 = 0$ and $y_2 - y_1 < 0$, then $\theta_P = \frac{1}{2}\pi$

These values are then normalised to a set number of *bits per value*, after which they are combined to create a hash value with the structure, $[H_1 : H_2 : H_3 : H_4]$. The resulting hash will have a size of $4 \times$ *bits per value* bits. Key points are only paired with smaller key points to ensure that the magnitude ratio is smaller than 1 and to remove any hash redundancy by not repeating key point pairs.

4.1.2 Database

After the set of hashes (fingerprint) has been calculated for a frame, the hashes are added to the database using a Video Frame Identification (VFID). The default amount of hashes created for a frame is 50. A VFID is a unique number each fingerprinted frame is given, to identify it within the database. The VFIDs are saved in a database in such a way that it can be retrieved very quickly when comparing unknown fingerprints to the database. For each hash value a specific number of spots are allocated for VFIDs within the database, as shown in Figure 4.2. The number of VFID spots shown in the figure is 16, but the default number of VFID spots is 64, but it can be changed if a larger database is required.

If a hash occurs in a frame, the frames' VFID is saved in one of that hash's VFID spots. The hashes are very unique, as there are 65536 if 16 bit hashes are used, so the spots will fill up slowly, but if all the spots of a hash are used, the adding of VFID information to that hash will be skipped. The database may be sized according to the requirement of the system by changing the amount of bits used to represent the VFIDs or by changing the number of available VFID spots per hash. The database will be optimised to use as little storage space as possible per frame fingerprint to allow it to contain as many video's fingerprints as possible. This optimisation will be done through the tests done in Chapter 5 and the results will explain why the default values mentioned in this paragraph where chosen.

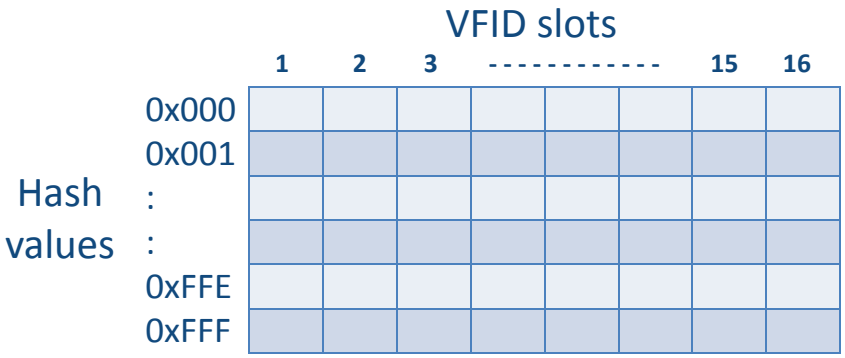

Figure 4.2: Diagram of database structure.

The video's name and frame number in the video is also saved along with the VFIDs in

a supplementary database, so the VFIDs is linked to meta data about the frame. This way, when a certain VFID is detected when querying a frame, the information about the originating video and frame number is available to use. The supplementary database consists of two tables. The first one is used to link the VFID to the VideoID and the position of the frame in the video. The second is used to link the VideoID to the video's name. This name will be displayed to the user, once the video is successfully detected.

4.1.3 Searching

To match an unknown frame's fingerprint to the database, the frame fingerprinting algorithm is applied to the frame to generate a set of hashes. All the VFIDs for every generated hash value are extracted from the database and every time a VFID occurs, it gets a point. After all the hashes' corresponding VFIDs have been accounted for, the VFID with the most accumulated points is the most credible match. If the number of hash matches exceed the match threshold it is seen as a definite match.

This searching technique is based on the Shazam algorithm's searching technique. The search is very quick and the speed stays more or less constant, regardless of the database's saturation [4]. The reason for this is that it doesn't have to do an exhaustive search by scanning through the entire database for a match, it only has to jump to the hashes that match exactly and extract its information. The maximum search time is a combination of the time it takes to read the VFIDs (*number of hashes* $\times$ *number of VFID spots*) and the time it takes to determine which VFID has the most hash matches.

4.2 Video Fingerprinting

To upgrade the frame detecting algorithm to a video detecting algorithm, a set of frames from the original video must be fingerprinted so they can be detected. Fingerprinting every single frame is very inefficient, because it takes up a lot of space in the database

and there are usually only small differences between consecutive frames in a video. To solve this problem a Key Frame Detector (KFD) is needed that can detect certain special frames in videos, for instance the first frames of shots or scenes. The important part of a KFD is its repeatability, meaning it should detect the same key frames in videos with the same content. KFDs have received a lot of attention in research and many attempts have been made to create a robust KFD [29] [28]. An existing video fingerprinting systems that uses a KFD is the one implemented by De Roover et al. in [9]. The use of key frames improve the fingerprinting system drastically, as it reduces the amount of data saved in the database as well as minimising processing time, because only a small percentage of the total frames in the video is fingerprinted and added to the database.

The video fingerprinting technique was developed with advertisement tracking as its main objective. For this a real time system is needed that can monitor a television station constantly, thus the KFD must be able to detect key frames in real time, meaning the processing time for each frame must be less than the frame refresh rate. In Figure 4.3 a basic diagram of the video fingerprinting system, using the KFD, is shown.

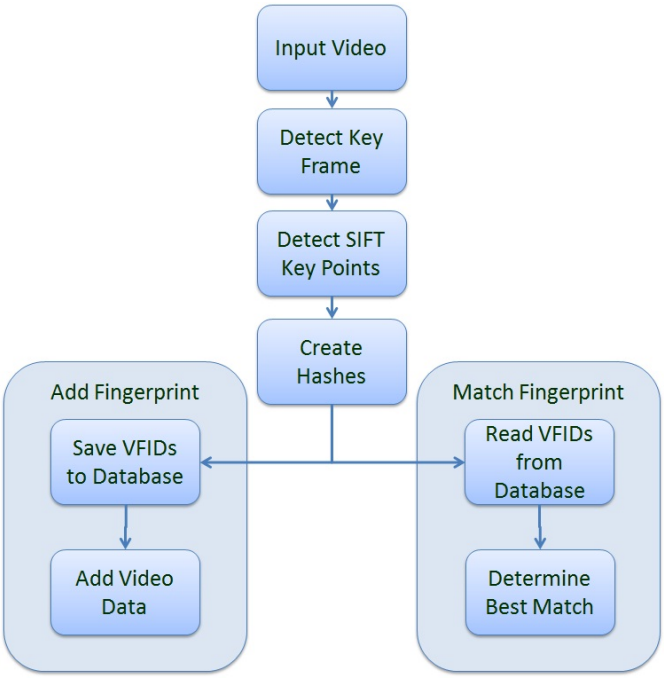

Figure 4.3: Diagram of the video fingerprinting system.

4.2.1 Key Frame Detector

There are a lot of existing video Shot Boundary Detectors (SBDs) and KFDs these days, but none of them retrieve shots and key frames perfectly. For this research a KFD is needed that can run in real-time and it should have a high recall rate, even with some distortion or codecs changes present. A good Frame Difference Measurement (FDM) to use in this instance would be SIFT, as used in the KFD in [29]. The reason for this is that the fingerprinting algorithm also uses SIFT, but the KFD can't be run in real time and therefore it can not be used for detection in the system.

The key frames in this instance were detected by using Jensen Shannon Divergence (JSD), proposed by Xu et al. in [28]. To quantise the difference of consecutive frames, the JSD of the two frames are calculated. Spikes in the JSD data represent shot boundaries in the video and it can thus be used to split the video into its different shots. Once a shot is detected a key frame can be selected from that shot. The first frame of the shots is selected as the key frame. In [9] it is mentioned that the first frame can be unreliable, but as the focus of this fingerprinting system is currently on advertisement tracking on television it won't be a problem.

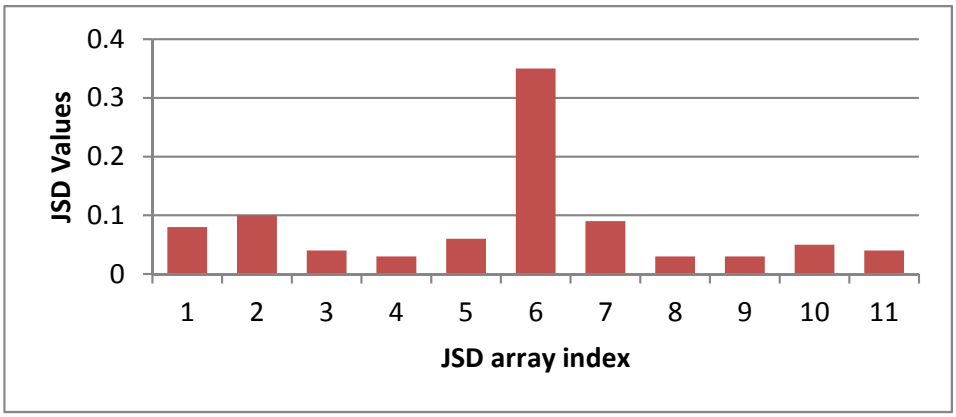

Figure 4.4: Example of JSD values containing a spike (shot boundary).

To detect a shot boundary, the spikes in the JSD data was detected by comparing a point

to the average of the points around it. If the point's value is bigger than the average, multiplied by a threshold, it is detected as a key frame. The threshold is equal to 3.5 and was determined experimentally. If the value is too low a lot of false shot boundaries are detected and if its too high a lot will be falsely missed. To implement the key frame detection using the JSD values, an array with a length of 11 is used to save the latest JSD values (see Figure 4.4). The number of JSD values to use was also determined experimentally. The following equations are then used with the array values to determine if the value in the 6th position (middle) is a spike:

If

$$JSD_6 > \sum_{i=1}^{5} JSD_i \div 5 \times 3.5 \tag{4.5}$$

and

$$JSD_6 > \sum_{i=7}^{11} JSD_i \div 5 \times 3.5 \tag{4.6}$$

The first equation takes the JSD values calculated for the five frames before the test frame, gets their average and multiplies it with the threshold of 3.5. The same is done in the second equation, but the JSD values are those of the five frames after the test frame. If the test frame's JSD is bigger than the first and seconds equation's answer, it is considered a key frame.

4.2.2 Video detection

To match a video to the database, the unknown video stream is analysed for key frames. Once a key frame is found, the frame is processed and its fingerprint is compared to the database. If the frame is matched to the database and the number of hashes matched is greater than 20%, the single frame detection will be enough to confirm a video detection, but if the number of hashes are between 10% and 20%, two frame matches are needed that correspond to the same video, to confirm the video match. The number of hashes matches needed for a positive match is discussed in Subsection 5.2.2.

4.3 Matlab

Matlab is a computing environment that allows a user to program prototype, mathematics-based algorithms, using the high-level programming language, and then test the algorithm easily. Matlab has a lot of built-in functions and classes (toolboxes) that can be used to quickly start working on a problem.

To test the idea for the fingerprinting system, it was implemented in Matlab, making use of the built-in image and video processing functions. To calculate the SIFT key points in the images and video frames, the VLFeat implementation of SIFT was used, that was programmed by Andrea Vedaldi [34]. The implementation was only done to determine the validity of the idea and after a few quick tests, it looked promising. In Matlab it is very easy to debug a program and see results, thanks to the user friendly Graphical User Interface (GUI) and visualisation tools. None of the early tests and results are shown in this dissertation, because it was only done as a proof of idea.

After the successful implementation in Matlab, it needed to be transferred to another programming language, so the system could be implemented in an environment with less overhead, thus improving processing speed of the algorithm.

4.4 Visual Basic (VB)

The final implementation was done in VB and it was used to run all the tests on the system. The system makes use of EmguVC, which is a wrapper for OpenCV in the C# and VB environments. OpenCV (Open Source Computer Vision) is an image processing library that is implemented in the C programming language. This is a very handy library for anyone who is working on video and image processing in the C# or VB environment and was used because it includes functions to extract SIFT and SURF key points, open and read videos, apply transformations to images and save images and videos. The implementations of SIFT is not exactly the same as explained in [1], but it is very close and thus adequate for the application.

The system makes use of a SQL server database to save all the relevant information of the fingerprints. SQL is a relational database management system and it is relatively easy to use in the VB environment, as libraries and support structures are included.