# Bibliography


[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, p. 91, Nov. 2004.

[2] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on DOI - 10.1109/ICCV.1999.790410*, vol. 2, 1999, pp. 1150–1157 vol.2.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 346–359, Jun. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2007.09.014

[4] A. Wang, "An industrial strength audio search algorithm," in *ISMIR*, 2003, pp. 7–13.

[5] (2012, August) http://www.kavonichone.co.za/blog/advertising/the-cost-of-tv-advertising/ 21-08-2012. http://www.kavonichone.co.za/blog/advertising/the-cost-of-tv-advertising/ 21-08-2012. [Online]. Available: http://www.kavonichone.co.za/blog/advertising/the-cost-of-tv-advertising/21-08-2012

[6] F. Galton, *Finger Prints*. McMillan & Co., London and New York, 1892.

[7] http://www.techrepublic.com/blog/tech-news/youtubes-video-fingerprinting-receives-mixed-reactions/1389. http://www.techrepublic.com/blog/tech-news/youtubes-video-fingerprinting-receives-mixed-reactions/1389. [Online]. Available: http://www.techrepublic.com/blog/tech-news/youtubes-video-fingerprinting-receives-mixed-reactions/1389

[8] D. Pereira and L. Loyola, "Robust video fingerprinting system," in *Communications and Electronics (ICCE), 2010 Third International Conference on*, 2010, pp. 141 –146.

[9] C. De Roover, C. De Vleeschouwer, F. Lefebvre, and B. Macq, "Robust video hashing based on radial projections of key frames," *Signal Processing, IEEE Transactions on*, vol. 53, no. 10, pp. 4020–4037, 2005.

[10] S. Lee and C. Yoo, "Video fingerprinting based on centroids of gradient orientations," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on DOI - 10.1109/ICASSP.2006.1660364*, vol. 2, 2006, pp. II–II.

[11] L. Sunil and C. Yoo, "Robust video fingerprinting for content-based video identification," *Circuits and Systems for Video Technology, IEEE Transactions on DOI - 10.1109/TCSVT.2008.920739*, vol. 18, no. 7, pp. 983–988, 2008.

[12] X. Su, T. Huang, and W. Gao, "Robust video fingerprinting based on visual attention regions," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on DOI - 10.1109/ICASSP.2009.4959886*, 2009, pp. 1525–1528.

[13] A. Ferman, A. Tekalp, and R. Mehrotra, "Robust color histogram descriptors for video segment retrieval and identification," *Image Processing, IEEE Transactions on DOI - 10.1109/TIP.2002.1006397*, vol. 11, no. 5, pp. 497–508, 2002.

[14] A. Massoudi, F. Lefebvre, C.-H. Demarty, L. Oisel, and B. Chupeau, "A video fingerprint based on visual digest and local fingerprints," in *Image Processing, 2006 IEEE International Conference on DOI - 10.1109/ICIP.2006.312834*, 2006, pp. 2297–2300.

[15] C. Kim and B. Vasudev, "Spatiotemporal sequence matching for efficient video copy detection," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 15, no. 1, pp. 127–132, 2005.

[16] R. Sun, X. Yan, and Z. Ding, "Robust image hashing using locally linear embedding," in *Computer Science and Service System (CSSS), 2011 International Conference on*, 2011, pp. 715–718.

[17] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on DOI - 10.1109/34.730558*, vol. 20, no. 11, pp. 1254–1259, 1998.

[18] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *SCIENCE*, vol. 290, pp. 2323–2326, 2000.

[19] D. Ellis. (2009) Robust landmark-based audio fingerprinting. [Online]. Available: http://labrosa.ee.columbia.edu/matlab/fingerprint/

[20] D. Lowe, "Local feature view clustering for 3d object recognition," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on DOI - 10.1109/CVPR.2001.990541*, vol. 1, 2001, pp. I–682–I–688 vol.1.

[21] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[22] R. Lienhart, "Comparison of automatic shot boundary detection algorithms," in *In Storage and Retrieval for Image and Video Databases, No. SPIE 3656. January 1999*, 1999, pp. 290–301.

[23] J. S. Boreczky and L. A. Rowe, "Comparison of video shot boundary detection techniques," in *Journal of Electronic Imaging*, 1996, pp. 170–179.

[24] M. Ahmed, A. Karmouch, and S. Abu-Hakima, "Key frame extraction and indexing for multimedia databases," in *Vision Interface '99, Trois-Rivires, Canada*, 1999, pp. 1–1.

[25] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proceedings of the third ACM international conference on Multimedia*. San Francisco, California, United States: ACM, 1995, pp. 189–200.

[26] F. Dirfaux, "Key frame selection to represent a video," in *Image Processing, 2000. Proceedings. 2000 International Conference on*, vol. 2, 2000, pp. 275–278 vol.2.

[27] D. Zhang, W. Qi, and H. J. Zhang, "A new shot boundary detection algorithm," in *Lecture Notes in Computer Science, 2195:63.* Springer-Verlag, 2001, pp. 63–70.

[28] Q. Xu, P. Wang, B. Long, M. Sbert, M. Feixas, and R. Scopigno, "Selection and 3d visualization of video key frames," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, 2010, pp. 52–59.

[29] G. Liu, X. Wen, W. Zheng, and P. He, "Shot boundary detection and keyframe extraction based on scale invariant feature transform," in *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, 2009, pp. 1126–1130.

[30] M. Cooper, J. Foote, J. Adcock, and S. Casi, "Shot boundary detection via similarity analysis," in *in Proceedings of the TRECVID 2003 Workshop*, 2003, pp. 79–84.

[31] H. Feng, W. Fang, S. Liu, and Y. Fang, "A new general framework for shot boundary detection and key-frame extraction," in *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval.* Hilton, Singapore: ACM, 2005, pp. 121–126.

[32] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 27, 1948.

[33] T. Cover and J. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications and Signal Processing. John Wiley & Sons, 2006. [Online]. Available: http://books.google.co.za/books?id=EuhBluW31hsC

[34] A. Vedaldi, "An open implementation of the sift detector and descriptor," 2007.

[35] C. E. Metz, "Basic principles of roc analysis," *Seminars in nuclear medicine*, vol. 8, no. 4, pp. 283–298, Oct. 1978. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/112681

[36] www.cse.msu.edu/ cse808/note/lecture11.ppt. www.cse.msu.edu/ cse808/note/lecture11.ppt. [Online]. Available: www.cse.msu.edu/~cse808/note/lecture11.ppt

[37] R. G. Sargent, "Verification and validation of simulation models," in *Proceedings of the 37th conference on Winter simulation*. Orlando, Florida: Winter Simulation Conference, 2005, pp. 130–143.

[38] A. M. Law and W. D. Kelton, *Simulation modeling and Analysis*. McGraw-Hill, New York, 1991.

# Appendix A

# Visual Basic Code

The following code is the code used to create the hashes from the list of key points:

```
' Take one input image and calculate its corresponding fingerprinting hashes
    Public Function CreateHashes(ByVal grayFrame As Image(Of Gray, Byte), ByVal startOctave As Integer) _
        As List(Of UInteger)
        'Create SIFT detector object
        Dim detector As New SIFTDetector(7, 3, startOctave, SIFTDetector.AngleMode.AVERAGE_ANGLE, 0.05, _
            10, 3, True, True)

        ' Resize if needed
        If (grayFrame.Width <> 320) And (grayFrame.Height <> 240) Then
            grayFrame = grayFrame.Resize(320, 240, INTER.CV_INTER_LINEAR)
        End If

        ' Get key points for each frame
        Dim keyPoints As MKeyPoint() = detector.DetectKeyPoints(grayFrame, Nothing).ToArray()

        ' Set the resolution of the hashes
        Dim resolution As Integer = CInt(Math.Pow(2, Parameters.NumBitsPerHash))

        ' Create Shazam-like hashes
        Dim magRel As Integer       ' relative magnitude
        Dim dirRel As Integer       ' relative direction of keypoint
        Dim pointDist As Integer    ' relative distance to keypoint
        Dim pointDir As Integer     ' relative direction to keypoint
        Dim tempValue As Double     ' a temporary value

        ' List of newly create hashes
        Dim newHashes As New List(Of UInteger)()
```

```
Dim enoughHashes As Boolean = False
Dim tier1 As Integer = 0, tier2 As Integer = 0

' While the number of hashes created has not reached the maximum
' or run out of key points
While Not enoughHashes
    ' Count through the current tier of key points of the first frame
    For count1 As Integer = tier1 * 10 To (tier1 + 1) * 10 - 1

        ' Break from the loop if the key points have run out
        ' and set the bool value to true to prevent and infinite loop
        If count1 >= keyPoints.Length Then
            enoughHashes = True
            Exit For
        End If
        If enoughHashes Then
            Exit For
        End If

        ' Get the four values of the key point
        Dim x1 As Double = keyPoints(count1).Point.X
        Dim y1 As Double = keyPoints(count1).Point.Y
        Dim sc1 As Double = keyPoints(count1).Size
        Dim th1 As Double = keyPoints(count1).Angle / 180 * Math.PI

        ' Count through the current tier of key points of the second frame
        For count2 As Integer = tier2 * 10 To (tier2 + 1) * 10 - 1

            ' Break from the loop if the key points have run out
            If count2 >= keyPoints.Length Then
                Exit For
            End If

            ' Get the four values of the key point
            Dim x2 As Double = keyPoints(count2).Point.X
            Dim y2 As Double = keyPoints(count2).Point.Y
            Dim sc2 As Double = keyPoints(count2).Size
            Dim th2 As Double = keyPoints(count2).Angle / 180 * Math.PI

            ' Calculate x and y difference between key point positions
            Dim xDif As Double = x2 - x1
            Dim yDif As Double = y2 - y1

            ' Calculate the distance between the key points in terms of pixels
            Dim dist As Double = Math.Sqrt(xDif * xDif + yDif * yDif)
```

```
' Calculate the values that make up the hash value
' and normalize them to fit into the set number of bits per hash
' if the key points are within a certain distance from each other
' and the first key point is bigger than the second one
' (This is important for the hashing of a single frame,
' so the same key points wont be used twice to create hashes)
If sc1 > sc2 Then
    If dist < sc1 Then
        If dist > 0.1 * sc1 Then

            ' Calculate relative magnitude
            magRel = CInt(Math.Floor((sc2 / sc1) * resolution))

            ' Calculate relative direction

            Dim relDirection As Double = th2 - th1
            While relDirection < 0 Or relDirection > 2 * Math.PI
                relDirection = CDbl((relDirection + 2 * Math.PI) Mod (2 * Math.PI))
            End While

            dirRel = CInt(Math.Floor(relDirection / (2 * Math.PI) * resolution))

            ' Calculate point distance

            pointDist = CInt(Math.Floor((dist - 0.1 * sc1) / (0.9 * sc1) * resolution))

            ' Calculate point direction

            If xDif = 0 Then
                If yDif > 0 Then
                    tempValue = (1 \ 2) * Math.PI
                Else
                    tempValue = (3 \ 2) * Math.PI
                End If
            Else
                tempValue = Math.PI - Math.Atan2(yDif, xDif)
            End If

            ' Normalise point direction
            tempValue = tempValue - th1

            ' Make sure value is between 0 and 2pi
            While tempValue < 0 Or tempValue > 2 * Math.PI
                tempValue = CDbl((tempValue + 2 * Math.PI) Mod (2 * Math.PI))
            End While

            pointDir = CInt(Math.Floor(tempValue / (2 * Math.PI) * resolution))
```

```
                    ' Check that the calculated values are within the set values
                    If magRel >= resolution Then
                        magRel = resolution - 1
                    End If
                    If pointDist >= resolution Then
                        pointDist = resolution - 1
                    End If
                    If pointDir >= resolution Then
                        pointDir = resolution - 1
                    End If
                    If dirRel >= resolution Then
                        dirRel = resolution - 1
                    End If


                    ' Create the hash code
                    Dim hash As UInteger = CreateHashValue(CByte(magRel), CByte(dirRel), _
                        CByte(pointDist), CByte(pointDir))

                    ' Check that the hash code didn't overflow max hash value
                    If hash > Parameters.TotalNumberOfHashes Then
                        hash = Parameters.TotalNumberOfHashes
                    End If

                    ' Add hash to the new hash list
                    newHashes.Add(hash)

                    ' Break if the new hash list reached the max hashes per frame
                    If newHashes.Count >= Parameters.MaxHashesPerFrame Then
                        enoughHashes = True
                        Exit For
                    End If
                End If
            End If
        End If
    Next
Next

' Count through tiers
' Tiers will count as follows: 0-0, 0-1, 1-1, 0-2, 1-2, 2-2, 0-3, 1-3, 2-3, 3-3 ....
If tier1 = tier2 Then
    tier1 = 0
    tier2 += 1
Else
    tier1 += 1
```

```
            End If
        End While

        ' Return the newly created hashes
        Return newHashes
    End Function
```

# Appendix B

# Conference Papers

### Video Fingerprinting using Robust Hashing

Presented at:
Southern African Telecommunication Networks and Applications Conference (SATNAC)
East London, Eastern Cape, South Africa
4 - 7 September 2011

### Video Fingerprinting Using Robust Hashing of Scale Invariant Features of Frames

Presented at:
Southern African Telecommunication Networks and Applications Conference (SATNAC)
George, Western Cape, South Africa
2 - 5 September 2012

# Video Fingerprinting using Robust Hashing

R. Moolman and W.C. Venter
School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus
Tel: +27 18 299 1961, Fax: +27 18 299 1977
Email: {20673892, willie.venter}@nwu.ac.za

***Abstract*—Video fingerprinting is a technique used to identify unknown video by matching the video to a known video which has similar content. This is achieved by creating a video fingerprint of the unknown video and comparing it to a database of known video fingerprints. There are many different video fingerprinting techniques that have been proposed, each with their strengths and weaknesses. In this paper a novel video fingerprinting technique is proposed. The technique is primarily designed for advertisement identification on a real-time video stream. For this purpose, robustness and speed are very important aspects of the algorithm. This fingerprinting technique is frame-based and makes use of the Scale Invariant Feature Transform algorithm and Shazam-like hashing between key points in different frames in a video sequence to create fingerprints.**

***Keywords:* video fingerprinting; automatic video recognition; perceptual frame hashing; content-based video identification; robust matching**

## I. Introduction

With the recent burst in media, it is very important to keep track of all the video content. Video fingerprinting is a technique that can be used to solve this problem and for this reason it has become very significant over the last few years. The main uses for video fingerprinting are copyright management and advertisement tracking [1][2]. There are many methods which already exist, but this research aims to expand what has been done in the field by developing a new technique focussed on speeding up the search process and improving on robustness.

In section II video fingerprinting, the Scale Invariant Feature Transform (SIFT) and Shazam's algorithm are briefly discussed, in section III a few existing video fingerprinting methods are mentioned, in section IV the proposed technique is explained and in section V a conclusion is laid out.

## II. Background

### A. Video Fingerprinting

In a video fingerprinting system a database of fingerprints is created from known videos. Once the database is set up, it can be used to identify unknown video by applying the fingerprinting algorithm to the unknown video and matching the query fingerprint to the database. Only a few seconds of the unknown video is required to find a match in the database. The quality of the video doesn't have to be perfect as the fingerprinting system allows for a number of distortions to be present whilst still getting a match.

A video fingerprinting system, as seen in Figure 1, consists of two major parts, namely the algorithm and the database. The algorithm is the video processing that is done on the video to extract useful information and create a fingerprint. The way the algorithm creates a fingerprint is very important as it influences the robustness, accuracy, efficiency and speed of the whole system. The database setup is of equal importance as it influences the search speed and storage space of the video fingerprints. It is important to represent the video fingerprint in a suitable fashion to improve on these aspects.

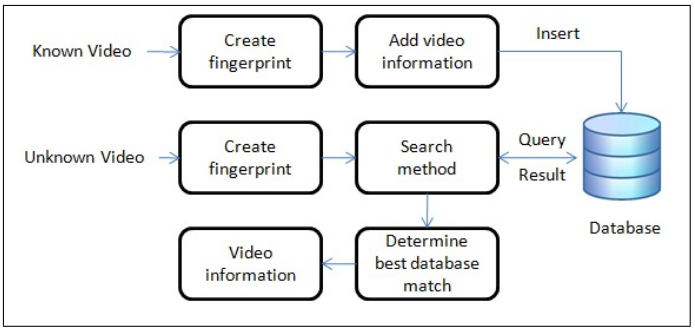



**Figure 1:** Diagram of a video fingerprinting system

### B. SIFT

The proposed algorithm makes use of the SIFT technique to calculate key points in frames. SIFT was designed by David Lowe for object recognition in pictures [3][4][5]. It was designed to handle scale and rotation variations, making it robust to common differences in different versions of the same video. Key points can also be produced in a short time for a picture or frame. SIFT was chosen to be used in the algorithm for it's speed and robustness.

The SIFT algorithm uses the Difference of Gaussian (DoG) scale space and gradients to find key points in a picture. A key point has four values: x-axis position, y-axis position, magnitude and direction. The full SIFT implementation uses these key points and forms a descriptor for each one by using 16 points around the key point and normalising their magnitude and direction against the key point, thus making them rotation and scale invariant.

### C. Shazam's algorithm

Shazam is an audio fingerprinting technique developed by Avery Wang [6]. The technique calculates the spectrogram of a given audio signal, after which key points are then found on the spectrogram where significant peaks form. Every key point has two values: time and frequency. Hashes are then created by calculating the beginning time, time difference and frequency difference between two key points and combining them into a hash code. A key point is only used to create a hash if it is within a certain region relevant to the main key point. A combination of hashes is the fingerprint for an audio signal.

## III. Existing Techniques

There are a few existing algorithms used in video fingerprinting. Companies like YouTube, the internet giant, have been utilizing this technology to manage copyrighted material since October 15, 2007 [7]. Others are also using it and the technology is gaining popularity very fast.

The most relevant existing systems are - Gradient of Centroids [8][9], Visual Attention Areas [2], Average Luminance over Time [1], Colour Histograms [10] and Visual Digest of Local Fingerprints [11]. Almost all of the existing techniques focus on the robustness and accuracy of the video fingerprinting, but seem to neglect the speed of the video fingerprint matches. This being said, the research proposed in this paper will focus on developing an algorithm that is not only robust, but fast; for the practical purpose of advertisement tracking in real-time video.

## IV. Technique

This video fingerprinting algorithm uses SIFT key points [3], for their robustness and the ability to create scale and rotation invariant hashes from the key points. The hashing method used is very similar to the one used in Shazam [6]. The proposed algorithm, database and searching method will be discussed in the section below.

### A. Algorithm

First, the video is sampled and each frame normalised by re-sampling it to a set resolution, after which the frames are converted to grey-scale. Key points for each frame are calculated using the SIFT algorithm. The key points with the largest magnitudes are used to create the hashes.

To create the hashes, the key points of two frames, a second apart, are used. Each key point in the first frame is matched with the key points in the second frame that are within the specified region around the key point. The x-axis pixel difference, the y-axis pixel difference, the magnitude difference and the direction difference of the key points are calculated and normalised to 6 bits per value, after which the four values are then combined to create a 24 bit hash value. Extra data is also added to link the hash to a specific frame and video. The hashes are then saved in the database.

Note that the number of bits used per value is a trade-off between accuracy and robustness. If the values are 8 bits the hashes will have a higher resolution and thus the number of hash matches in a search will decrease while increasing the accuracy of a match, but the robustness will decrease as small changes may cause hash matches to be missed. If 4 bits are used the number of false positives will increase, but the robustness will also be greatly increased, for small errors are now allowed. Thus, a good balance between the two requirements is met with 6 bits per value, which is used in this algorithm. Further research may justify changes that result in a better balance.

### B. Database

The database will be optimised to use as little storage space as possible whilst providing a fast searching method. This is an important aspect as the information of all the video's fingerprints is quite large.

A Video Frame Identification (VFID) is a number each frame in every fingerprinted video is given to identify it with. It is these VFIDs that are saved in a database in such a way that it can be retrieved very quickly. For each hash value a specific number of spots are allocated for VFIDs within the database. If a hash occurs in a frame, the frames' VFID is saved in one of that hash's spots. The hashes are very unique, so the spots will fill up slowly. 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.

### C. Searching

To match a frame's fingerprint to the database, the video fingerprinting algorithm is applied to the frame to generate a set of hashes. 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 above a specified threshold is the most credible match.

This searching technique is based on the Shazam algorithm's searching technique as the search is very quick and the speed stays more or less constant even when the database becomes really large [6]. The reason for this is that it doesn't have to scan through the entire database for a match, it only has to jump to the exactly matching hashes and extract its information.

## V. Conclusion

In this paper a new method for video fingerprinting was proposed, based on the combination of the SIFT and Shazam algorithms. The purpose of this research it to determine whether this algorithm is practical or not. The algorithm has the capability to be implemented using the Graphics Processing Unit (GPU) for further speed advantages. Based on the background, the algorithm deems promising but it is only a hypothesis and still needs to be implemented and tested to produce the desired results.

## References


[1] D. Pereira and L. Loyola, "Robust video fingerprinting system," in *Communications and Electronics (ICCE), 2010 Third International Conference on*, pp. 141 –146, 2010.

[2] X. Su, T. Huang, and W. Gao, "Robust video fingerprinting based on visual attention regions," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on DOI - 10.1109/ICASSP.2009.4959886*, pp. 1525–1528, 2009.

[3] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on DOI - 10.1109/ICCV.1999.790410*, vol. 2, pp. 1150–1157 vol.2, 1999.

[4] D. Lowe, "Local feature view clustering for 3d object recognition," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on DOI - 10.1109/CVPR.2001.990541*, vol. 1, pp. I–682–I–688 vol.1, 2001.

[5] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, p. 91, Nov. 2004.

[6] A. Wang, "An industrial strength audio search algorithm," in *ISMIR*, pp. 7–13, 2003.

[7] "http://www.techrepublic.com/blog/tech-news/youtubes-video-fingerprinting-receives-mixed-reactions/1389."

[8] S. Lee and C. Yoo, "Video fingerprinting based on centroids of gradient orientations," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on DOI - 10.1109/ICASSP.2006.1660364*, vol. 2, pp. II–II, 2006.

[9] S. Lee and C. Yoo, "Robust video fingerprinting for content-based video identification," *Circuits and Systems for Video Technology, IEEE Transactions on DOI - 10.1109/TCSVT.2008.920739*, vol. 18, no. 7, pp. 983–988, 2008.

[10] A. Ferman, A. Tekalp, and R. Mehrotra, "Robust color histogram descriptors for video segment retrieval and identification," *Image Processing, IEEE Transactions on DOI - 10.1109/TIP.2002.1006397*, vol. 11, no. 5, pp. 497–508, 2002.

[11] A. Massoudi, F. Lefebvre, C.-H. Demarty, L. Oisel, and B. Chupeau, "A video fingerprint based on visual digest and local fingerprints," in *Image Processing, 2006 IEEE International Conference on DOI - 10.1109/ICIP.2006.312834*, pp. 2297–2300, 2006.



**R. Moolman** is currently pursuing a Master's degree in Computer and Electronic Engineering at the North-West University. He received his Bachelor's degree in Computer and Electronic Engineering in 2010. His current research interests are video fingerprinting, digital signal processing and software design.

# Video Fingerprinting Using Robust Hashing of Scale Invariant Features of Frames

R. Moolman and W.C. Venter
School of Electrical, Electronic and Computer Engineering
North-West University, Potchefstroom Campus
Tel: +27 18 299 1961, Fax: +27 18 299 1977
Email: {20673892, willie.venter}@nwu.ac.za

***Abstract*—Video fingerprinting is a technique used to identify an unknown video by matching the video's fingerprint to a database of video fingerprints. The fingerprints are derived through algorithms and characterizes the video's content. Video fingerprinting automates video identification and it has many uses these days with the exponential increase in video usage. A high quality video fingerprinting system should be fast, robust to distortions and use storage space efficiently. All these aspects are dependant on the technique used to fingerprint the videos. In this paper a novel video fingerprinting technique is proposed. This technique was developed with the intention to detect videos in real time, with its primary use being advertisement tracking. The technique makes use of the Scale Invariant Feature Transform (SIFT) algorithm, combined with the idea of hashing introduced by Shazam, an audio fingerprinting technique, to fingerprint and detect single frames. The aspects of the frame detector are discussed and tested, after which the optimal variable values are chosen for frame detection.**

***Keywords:* video fingerprinting; copy detection; automatic video recognition; perceptual frame hashing; content-based video identification; robust matching**

## I. Introduction

In recent years the use of video has increased dramatically. Video fingerprinting is a technique that can be implemented to automate the management of this large amount of video data. For this reason it has received a lot of attention in research over the last few years. Video fingerprinting can be used for copyright management [1], advertisement tracking [2][3] and video management on personal systems.

In the video detection system a known video is run through an algorithm to extract its fingerprint and then it is saved to the database. When detecting an unknown video, the same algorithm is used to fingerprint the video, except this time the fingerprint is compared to the database to find the best match. If the video's fingerprint was added to the database, the unknown video should be detected. As all video fingerprinting systems aim for the same goal, there are certain traits that are expected of the system, namely:

- **Robustness**: The ability to detect videos while distortion is present.
- **Accuracy**: The ability to give unique fingerprints to different videos.
- **Speed**: The ability to fingerprint and detect videos quickly.
- **Efficiency**: The ability to effectively store and match fingerprint data in a database.

A lot of previous work has been done in this field, including [4], which uses Radial Projection of key frames, Gradient of Centroids [5][6], Visual Attention Areas [3], Average Luminance over Time [2], Colour Histograms [7] and Visual Digest of Local Fingerprints [8]. Most of the existing techniques focus on the robustness and accuracy of the video fingerprinting, but seem to neglect the speed of the video fingerprint matches. This research aims to expand what has been done in the field by developing a new technique that focuses on speed, while maintaining robustness, accuracy and efficiency.

In Section II the technique is explained in detail, and in Section III video detection is discussed. Section III is separate because it makes use of a basic key frame detector and the frame fingerprinting to detect videos. Sections II and III both include subsections where tests are discussed and results are shown. Lastly, Section IV contains a conclusion along with proposed future work.

## II. Frame Fingerprinting

While creating a robust frame detection algorithm to be used in a video fingerprinting system, 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.

Shazam is an audio fingerprinting technique that makes use of hashes, created from pairs of key points found on the song's spectrogram, to quickly match an unknown audio file to the database [9]. To utilize this technique to detect frames as fast as possible, an algorithm called SIFT was used. SIFT detects robust, scale and rotation invariant key points in an image and was originally designed for object recognition [10]. Each SIFT key point consists of a x and y coordinate, a magnitude and a direction. By using the key points detected by the SIFT algorithm, Shazam-like hashes can be created to fingerprint the frames. The proposed algorithm, database and searching method will be discussed in the section below.

### A. Algorithm

First, the video is sampled and each frame normalised by re-sampling it to a set resolution of $320 \times 240$, after which the frames are converted to gray-scale. Key points for the frame are then calculated using the SIFT algorithm. The key points with the largest magnitudes are used to create the hashes, 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 detected.

Each key point in the frame is matched with other key points found in the frame that are within the specified region

around the key point. The values of the magnitude ratio, the direction difference, relative distance and direction to the secondary key point are then calculated and normalised to a set number of bits per value, after which they are then combined to create a hash value. Key points are only matched 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. Once a certain number of hashes per frame is calculated, the hashing ceases. This hash limit is discussed in Section II-D2.

### B. Database

A Video Frame Identification (VFID) is a number each fingerprinted frame of a video is given, to identify it with. It is these VFIDs that are saved in a database in such a way that it can be retrieved very quickly. For each hash value a specific number of spots are allocated for VFIDs within the database. If a hash occurs in a frame, the frames' VFID is saved in one of that hash's spots. The hashes are very unique, so the spots will fill up slowly. 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.

Video and frame data is also saved with the VFIDs in a supplementary database, so the VFIDs carry meaning. 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.

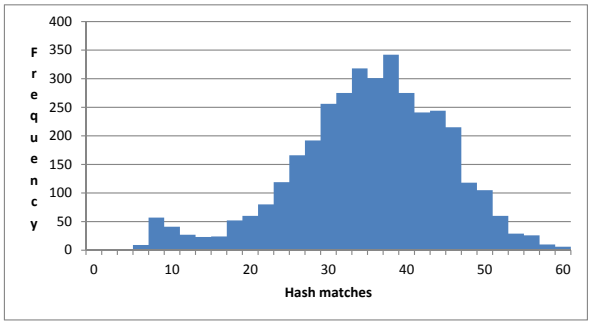
### C. Searching

To match a frame's fingerprint to the database, the video 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 above a specified threshold is the most credible match.

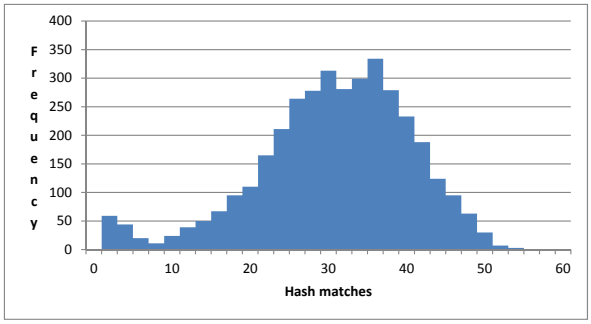
This searching technique is based on the Shazam algorithm's searching technique. The search is very quick and the speed stays more or less constant even when the database becomes really large [9]. The reason for this is that it doesn't have to do an exhaustive search by scanning through the entire database 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 (maximum of $hashes \times number\ of\ VFID\ slots$) and the time it takes to determine which VFID has the most hash matches.

### D. Tests

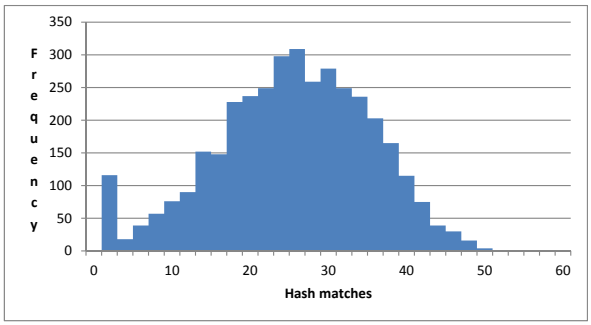
The frame fingerprint consists of a set of hashes, that is created from key points, as mentioned above. While comparing an unknown frame to the database, the hashes created for the frame should match exactly to the hashes created from the original frame for it to be detected as a match, even if there are distortions present. The variables that affect detection are the number of bits used for a hash, the number of hashes used to fingerprint a frame and the detection threshold. Each of these variables will be discussed and the tests done, to determine the best value for each variable, will be covered in this subsection.




(a) Frequency histogram of hash matches for AVI XviD using 12 bit hash



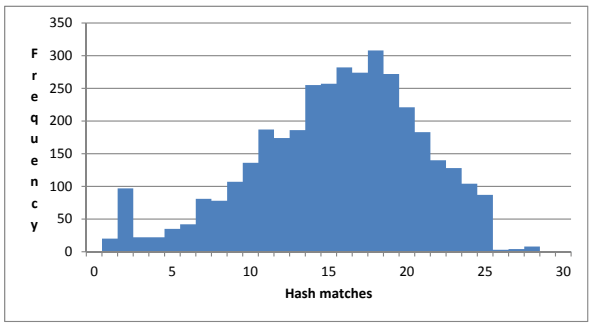

(b) Frequency histogram of hash matches for AVI XviD using 16 bit hash



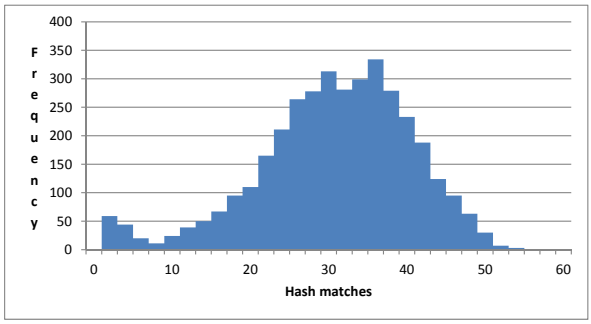

(c) Frequency histogram of hash matches for AVI XviD using 20 bit hash

**Figure 1:** Results for *Bits per Hash* test

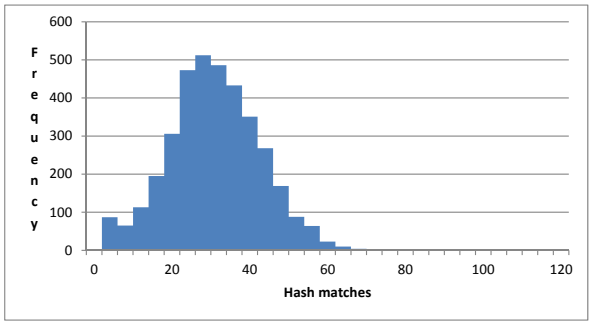
*1) Bits per Hash:* Each hash value consists of four variables normalised to use only a certain amount of bits. The number




(a) Frequency histogram of hash matches for AVI XviD using 25 hashes




(b) Frequency histogram of hash matches for AVI XviD using 50 hashes




(c) Frequency histogram of hash matches for AVI XviD using 100 hashes

**Figure 2:** Results for *Detection Threshold and Hashes per Frame* test

of bits used for each hash is important, because hash values have to be exactly the same to get a match with the database. The higher the number of bits used, the higher the resolution, thus the hashes are more specific and its harder to get matches. On the other hand, using less bits to create hashes, means that its easier to get matches, but it increases the chances of false matches. It also increases the change of getting two or three of the same hash values in one frame.

The number of bits used per hash value is a trade-off between accuracy and robustness. Using more bits per hash will result in a higher hash resolution and thus increase the accuracy of a match, but the robustness will decrease as small changes may cause hash matches to be missed, resulting in false negative frame matches. On the other hand, if less bits are used the number of false positive matches will increase, but the robustness will also be greatly increased, because small errors will be allowed.

To determine the optimal number of bits to use for creating hashes, the following test was done: Firstly, the number of hashes per frame were set to 50. Then a set of 30 music videos were fingerprinted and their fingerprints saved to the database using a specific number of bits per hash. Music videos were used because of the wide array of images and scenes used in the videos. Once the database was created, the videos were converted to other formats (using FormatFactory) and run through the detection algorithm. In Figure 1 frequency histograms for the number of hashes matched are shown. The frequency histograms are made up of 3687 frame match's data. The test also included determining the second best match (first false positive) to frame, to determine the usual number of matches a frame gets if it's not in the database, but the data is only mentioned in Table I.

| Figure | Best match average | Best match standard deviation | Second match average | Second match standard deviation |
|---|---|---|---|---|
| 1(a) | 34.83 | 10.02 | 8.51 | 2.6 |
| 1(b) | 30.31 | 9.85 | 2.81 | 1.97 |
| 1(c) | 24.93 | 9.9 | 1.3 | 1.38 |

Table I
QUANTITIVE RESULTS FOR BITS PER HASH TEST

If one looks at the graphs in Figure 1 one can see that the data distribution is almost normal, except for the outliers at the very low values. These outliers are caused by the frame detector algorithm, because frames are sometimes detected as key frames in the avi files, while they weren't detected in the mp4 files.

If one compares the data seen in Table I, one can see that the 12 bit hashes have a higher match average than the 16 bit hashes, but the lower resolution of the 12 hash matches cause a big increase in the average of the secondary matches. In this case the 16 bit hashes prove advantageous, as it keeps the hash uniqueness higher (good for database) with only a little decrease in hash matches. The 20 bit hashes have lower averages than the 16 bit hashes, but both options are acceptable for implementation. The deciding factor is the balance between uniqueness, robustness and efficiency, and thus, the 16 bit hashes are better.

*2) Detection Threshold and Hashes per Frame:* Every single fingerprinting system makes use of thresholds to determine if the query fingerprint has a legitimate match in the database. This frame fingerprinting technique, makes use of a minimum

hash matches threshold to determine if the match is successful. If the best VFID match's number of matches is not above the threshold, it is not considered a match. Another variable that also affects the frame detector is the number of hashes created for each frame fingerprint. Both the number of hashes per fingerprint and the minimum hash match threshold is determined by the same set of test as the minimum hash match threshold is dependant on the number of frames used to fingerprint a frame.

To determine these thresholds, the following test was done: As with the bits per hash test, a set of music video's fingerprints were added to the database, using 25 as the maximum hash threshold. The test was repeated again two times while using 50 and 100 as the maximum hash threshold. The original videos that were fingerprinted and added to the database, used the mp4 wrapper and DivX codec, while the videos used to match to the database, are in avi wrappers with XviD encoding. In Figure 2, the results for the tests are shown in the form of frequency histograms. The frequency histograms are made up of 3716, 3687 and 3650 frame match's data, respectively. The differences are due to the fact that frames are only added to the database if their number of hashes exceed the maximum hash threshold. The test also included determining the second best match (first false positive) to frame, as with the bits per hash test and are included in Table II.

| Figure | Best match average | Best match standard deviation | Second match average | Second match standard deviation |
|---|---|---|---|---|
| 2(a) | 15.51 | 5.44 | 1.92 | 1.03 |
| 2(b) | 30.31 | 9.85 | 2.81 | 1.97 |
| 2(c) | 29.48 | 11.64 | 4.41 | 2.82 |

Table II
QUANTITIVE RESULTS FOR HASHES PER FRAME TEST

If the results in Table II are compared, it is clear to see that the average number of hash matches stay almost constant when using 100 hashes per frame instead of 50. The reason for this is that the biggest key points are the most robust, allowing the first 50 or so hashes to be the most repeatable. The test was also done with flv (flv1) and mpg (mpeg1) files and the results reflect the same information, although the data is not shown in this paper.

Detection was also tested while using 25 hashes per frame. The results show that the averages of the 25 hashes per frame are basically half of the 50 hashes per frame averages, thus 50 hashes per frame will be used to increase the chance of hash detection and allowing more space to choose a detection threshold.

After close inspection of the results, it was seen that frames that have more than 10% hash matches of the hashes used per frame are positive matches, almost 100% of the time. There may still be exceptions, so it is safe to say that the minimum hashes detection threshold can be set to 20%. So, for detection with 50 hashes per frame, any frame match that has more than 10 hash matches can be considered a positive match.

## III. VIDEO FINGERPRINTING

The video fingerprinting technique discussed in this paper was developed with advertisement tracking as its main objective. For this a real time system is needed to monitor television stations constantly.

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, so a key frame detector was needed that could detect the same frames in videos with the same content. Key frame detectors have received a lot of attention in research and many attempts have been made to create a robust key frame detector [11] [12]. An existing video fingerprinting systems that uses a key frame detector is [4]. 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. In Figure 3 a basic diagram is shown of the system.

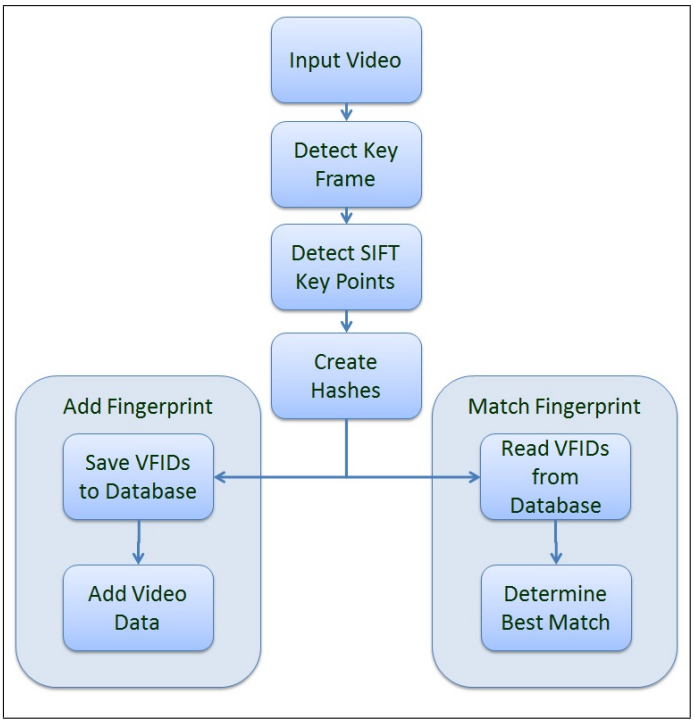



**Figure 3:** Diagram of a video fingerprinting system

### A. Key Frame Detector

There are a lot of existing video shot and key frame detectors these days, but none of them retrieve shots and key frames perfectly. For this research a key frame detector is needed that can be 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 to use in this instance would be SIFT, as used in the key frame detector in [11]. The reason for this is that the fingerprinting algorithm also uses SIFT, but the key frame detector 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 where detected by using Jensen Shannon Divergence (JSD), proposed by [12]. 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 [4] 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.

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 certain amount, it is detected as a key frame.

*Test:* Key frame detection is very important for the successful matching of a video's frames. The same frames have to be selected as key frames every time a video is run through the algorithm, so matches will be found in the database. In this section the robustness of the key frame detector is shown.

The recall and precision measurements are used to evaluate information retrieval systems, and can be defines as:

$$precision = \frac{t_p}{t_p + f_p} \tag{1}$$

$$recall = \frac{t_p}{t_p + f_n} \tag{2}$$

Where $t_p$ is the true positives (correct detections), $f_p$ the false positives (wrong detections) and $f_n$ the false negatives (missed detections). After running the test on the avi xvid files the following results were found:

| Wrapper | Codec | $t_p$ | $f_p$ | $f_n$ | Precision | Recall |
|---|---|---|---|---|---|---|
| avi | XviD | 3594 | 93 | 89 | 0.975 | 0.976 |
| mpg | mpeg1 | 3568 | 78 | 116 | 0.979 | 0.968 |

Table III
KEY FRAME TEST RESULTS

### B. 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 compared to the database. If the frame was detected successfully, and the number of hashes matched was greater than 30% the video is detected, but if the number of hashes are between 10% and 30%, two consecutive frame matches are needed to confirm a video match.

*Test:* To test the video detection, a day's video was recorded from a television channel. All the advertisements where then fingerprinted and added to the database using 50 hashes per key frame and 16 bit hashes. The video data was then run through the detector. Of the 128 advertisements, 124 were detected successfully, some being detected more than once, when they were repeated in other time slots. In Table IV the results are shown, with the first line only referring to the advertisements and if they were detected or not during the day, whereas the the second line represents all the advertisements shown during the day, repeats included.

| | $t_p$ | $f_n$ | $f_p$ | Precision | Recall |
|---|---|---|---|---|---|
| Advertisements | 124 | 4 | 0 | 1 | 0.969 |
| Advertisements (repeats included) | 188 | 9 | 0 | 1 | 0.954 |

Table IV
VIDEO FINGERPRINTING

The advertisements that were missed was because the key frame detector failed to detect sufficient key frames in each of the videos. This is because the key frame detector detects abrupt changes and these advertisements didn't have any.

## IV. CONCLUSION

In this paper a new method for video fingerprinting was proposed, based on the SIFT technique and a Shazam-like hashing method. The test results show the frame fingerprinting algorithms' detection characteristics and meaningful decisions was made that improves the detection effectiveness of the algorithm. A key frame detector algorithm is also used to use frame fingerprinting to in video fingerprinting. The system does function correctly and can detect videos, but future work will include improving the key frame detector and making the frame fingerprinting more robust towards other types of distortions.

## REFERENCES


[1] "http://www.techrepublic.com/blog/tech-news/youtubes-video-fingerprinting-receives-mixed-reactions/1389."

[2] D. Pereira and L. Loyola, "Robust video fingerprinting system," in *Communications and Electronics (ICCE), 2010 Third International Conference on*, pp. 141 –146, 2010.

[3] X. Su, T. Huang, and W. Gao, "Robust video fingerprinting based on visual attention regions," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on DOI - 10.1109/ICASSP.2009.4959886*, pp. 1525–1528, 2009.

[4] C. De Roover, C. De Vleeschouwer, F. Lefebvre, and B. Macq, "Robust video hashing based on radial projections of key frames," *Signal Processing, IEEE Transactions on*, vol. 53, no. 10, pp. 4020–4037, 2005.

[5] S. Lee and C. Yoo, "Video fingerprinting based on centroids of gradient orientations," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on DOI - 10.1109/ICASSP.2006.1660364*, vol. 2, pp. II–II, 2006.

[6] S. Lee and C. Yoo, "Robust video fingerprinting for content-based video identification," *Circuits and Systems for Video Technology, IEEE Transactions on DOI - 10.1109/TCSVT.2008.920739*, vol. 18, no. 7, pp. 983–988, 2008.

[7] A. Ferman, A. Tekalp, and R. Mehrotra, "Robust color histogram descriptors for video segment retrieval and identification," *Image Processing, IEEE Transactions on DOI - 10.1109/TIP.2002.1006397*, vol. 11, no. 5, pp. 497–508, 2002.

[8] A. Massoudi, F. Lefebvre, C.-H. Demarty, L. Oisel, and B. Chupeau, "A video fingerprint based on visual digest and local fingerprints," in *Image Processing, 2006 IEEE International Conference on DOI - 10.1109/ICIP.2006.312834*, pp. 2297–2300, 2006.

[9] A. Wang, "An industrial strength audio search algorithm," in *ISMIR*, pp. 7–13, 2003.

[10] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, p. 91, Nov. 2004.

[11] G. Liu, X. Wen, W. Zheng, and P. He, "Shot boundary detection and keyframe extraction based on scale invariant feature transform," in *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, pp. 1126–1130, 2009.

[12] Q. Xu, P. Wang, B. Long, M. Sbert, M. Feixas, and R. Scopigno, "Selection and 3d visualization of video key frames," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*, pp. 52–59, 2010.



**R. Moolman** is currently pursuing a Master's degree in Computer and Electronic Engineering at the North-West University. He received his Bachelor's degree in Computer and Electronic Engineering in 2010. His current research interests are video fingerprinting, digital signal processing and software design.