# COPY DETECTION OF 3D VIDEOS

by

Naghmeh Khodabakhshi

B.Sc., Sharif University of Technology, 2010

a Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

## APPROVAL

| | |
|---|---|
| **Name:** | Naghmeh Khodabakhshi |
| **Degree:** | Master of Science |
| **Title of Thesis:** | Copy Detection of 3D Videos |

**Examining Committee:** Dr. Joseph G. Peters
Professor, Computing Science
Simon Fraser University
Chair

Dr. Mohamed Hefeeda,
Associate Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Jiangchuan Liu,
Associate Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Greg Mori,
Associate Professor, Computing Science
Simon Fraser University
Examiner

**Date Approved:** 16 July 2012

# Abstract

We present a novel content based copy detection system for 3D videos. The system creates compact and robust depth and visual signatures from 3D videos. The system returns a score, using both spatial and temporal characteristics of videos, indicating whether a given query video matches any video in a reference video database, and in case of matching, which portion of the reference video matches the query video. Our analysis shows that the system is efficient, both computationally and storage wise. The system can be used, for example, by video content owners, video hosting sites, and third-party companies to find illegally copied 3D videos. We implemented Spider, a complete realization of the proposed system, and conducted rigorous experiments on it. Our experimental results show that the proposed system can achieve high accuracy in terms of precision and recall even if copied 3D videos are subjected to several modifications at the same time. For example, the proposed system yields 100% precision and recall when copied videos are parts of original videos, and more than 90% precision and recall when copied videos are subjected to different individual modifications such as cropping, scaling, and blurring.

# Acknowledgments

It is my pleasure to express my profound gratitude to my supervisor, Dr. Mohamed Hefeeda, for his invaluable guidance, incessant encouragement, and persistent support throughout the course of this research. Without his critical reviews and intellectual inputs, completion of this thesis would not have been possible for me.

I would like to express my gratitude to Dr. Jiangchuan Liu, and Dr. Greg Mori, for being on my committee and reviewing this thesis. I also would like to thank Dr. Joseph Peters for taking the time to chair my thesis defense.

I am grateful to all the members at the Network Systems Lab for providing me a stimulating and fun environment. I would like to thank all my wonderful friends who comforted me during the difficult times, and offered me great support and help.

Finally, and most importantly, I would like to offer my endless gratitude to my family for their ceaseless love and support.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Overview

The technological progress over the past years has enabled the general public, who were at the consuming end in the past, to easily communicate via the Internet's new intelligent web services and spread their own generated content. Youtube, as the first mass-popular platform for user created media content, reported[1] that 60 hours of video were uploaded every minute, and they had 1 trillion views in 2011. Although this gives the users the ability to share their works and ideas, it also makes it relatively easy to copy commercial content and distribute it over the Internet. Copyright infringement of commercial videos is a serious issue that can result in loss of revenue for businesses.

Three dimensional (3D) video is becoming popular, as its technology is getting mature. Now, content creators are producing more materials in 3D formats, cinemas are upgraded to 3D, 3DTV broadcasting has become reality, and even video sharing websites, such as YouTube, provide means to upload and watch 3D contents. Many experts are anticipating that the future of video is 3D. With rapid advances in 3D cameras and displays, numerous 3D videos are expected to be created and consumed in the near future. Since the creation of 3D contents is expensive, content owners would be interested in protecting their contents from illegal copying and distribution, especially posting on online sites.

Watermarking and content based copy detection are two fundamental approaches to video copy detection. Content based copy detection, which is the focus of this thesis, is

[1]www.youtube.com/t/press_statistics

based on extracting distinctive features from the videos. These features are called signatures or fingerprints. If the signature of two videos are detected to be similar, then one of the videos may be a copy of the other.

## 1.2 Problem Statement

Detecting copies of a 2D video is a complex task. First, videos are composed of many frames (usually 25 or 30 frames per second), and comparing numerous frames from potential video copies against reference videos is computationally intensive. Detection of video copies is also complicated by the fact that many edit effects occur on the copied videos. These edits, usually called video *transformations*, can be done intentionally to avoid detection or unintentionally because of the copying process. Scaling, blurring, cropping, slow motion, and change of frame rate are examples of these transformations. In addition, there are some transformations, like synthesizing new views, which are specific to 3D videos, and make the copy detection even more challenging.

A good copy detection system will represent the video content by robust, discriminating, and compact signatures. A robust signatures remain recognizable in the face of edit effects. A discriminative signature is a good representer of a video content, but unlikely to be found in videos which have different content. A compact signature requires less storage space and less computation to evaluate the similarity between two signatures. The search process should also be efficient and fast to effectively determine copies in large databases.

The problem addressed in this thesis can be stated as follows:

**Problem** (3D Video Copy Detection)**.** *Find copies of a given 3D video in a large collection of videos, even if copies are modified and/or embedded in other videos.*

## 1.3 Thesis Contributions

The contributions of this thesis can be summarized as follows [18] [19]:

- We propose compact signatures for both texture and depth signal of 3D videos. The signatures are storage-efficient but still discriminating enough for good matching results.

- We present a novel and complete content based copy detection system for 3D videos. The system extracts compact and robust signatures from the 3D videos, both texture and depth. Then, these signatures are used to compare videos. Signatures of query videos are created online and compared against signatures of reference videos in the indexed database.
- We implemented Spider, a complete realization of the proposed system. The system provides users with a graphical user interface for easy setup and visualization [19].
- We performed rigorous evaluation of the Spider system considering various modifications to the test videos, in order to simulate realistic conditions for regular operation. 3D videos collected from different sources with diverse characteristics were used to evaluate the proposed system. Our experimental results show that the proposed system can achieve high accuracy in terms of precision and recall even if the 3D videos are subjected to several transformations at the same time. For example, the proposed system yields 100% precision and recall when copied videos are parts of original videos, and more than 90% precision and recall when copied videos are subjected to different individual transformations.

## 1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 provides background information needed to understand the concepts discussed in this thesis. Chapter 3 explains the proposed system to detect 3D videos copies. It describes all steps and components of the system, as well as its running time and memory analysis. The implementation details of the proposed system is explained in Chapter 4. Chapter 5 evaluates the proposed copy detection system and Chapter 6 provides conclusions as well as ideas for future work.

# Chapter 2

# Background and Related Work

## 2.1 Background

This chapter provides the background information needed to understand the concepts discussed in this thesis. We present an overview of the 3D video technology, and video copy detection concept. We also discuss current algorithms and techniques used in this research area.

### 2.1.1 3D Videos

Human eyes are horizontally separated by about 50-75 mm depending on each individual [45]. Consequently, each eye has a slightly different view of the world. At any given moment, the lines of sight of the two eyes meet at a point in space. This point in space, called fixation point, projects to the same location (i.e. the center) on the retina of the two eyes. However, many other points, as shown in Figure 2.1, do not fall on corresponding retinal locations because of different viewpoints of the left and right eyes. This difference between the points of projection in the two eyes is called binocular disparity. This disparity has inverse relation with the depth of the point, so human brain uses it to extract depth information.

Similarly, 3D video enables depth perception by providing two slightly different views for each eye of the user. Then, the human brain fuses these two views to perceive the depth of the video scene. Different methods exist for preparing 3D videos:

- Stereo Video: The video has two views. A view can be thought of as a separate 2D

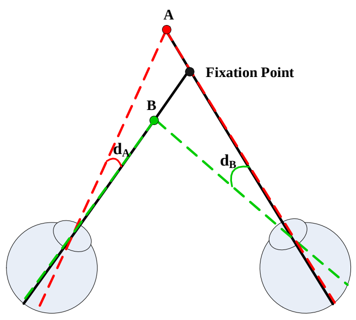


Figure 2.1: The fixation point projects to the same location in both eyes. However, other points like A and B project to different locations in left and right eyes. This difference ($d_A$ and $d_B$) are called disparity.

video stream.

- Multi-view Video: The video has multiple views and a subset of them is displayed to the user depending on the angle of viewing.
- Video plus Depth: In this case, the video is encoded in 2D and a separate depth map is created for the 2D video. The depth information of each frame is usually represented as a gray-level image showing the depth of each pixel in that video frame. An example of a video frame and its depth map is shown in Figure 2.2. The depth map allows the creation of many virtual (synthesized) views, which adds flexibility and support wider viewing angles for users. Creating of virtual views, however, is computationally expensive and could introduce some visual artifacts.

Combinations of the above methods are possible, as described in [17] and [26]. For example, a 3D video can be encoded in multi-view plus depth, where a few views are used with the depth map to create more virtual views.

When the 3D videos are created, they need to be compressed, transmitted, and displayed. Different coding standards have been developed for compression and transmission of 3D videos, but it is still an ongoing process. In some coding standards, like H.264/AVC simulcast, video sequences for each view are individually and independently encoded, transmitted, and decoded. While this approach is simple, it is highly redundant, since the inter-view dependencies are ignored. On the other hand, coding standards like H.264/AVC SEI and H264/MVC exploit the inter-view dependencies for better compression [27] [26].

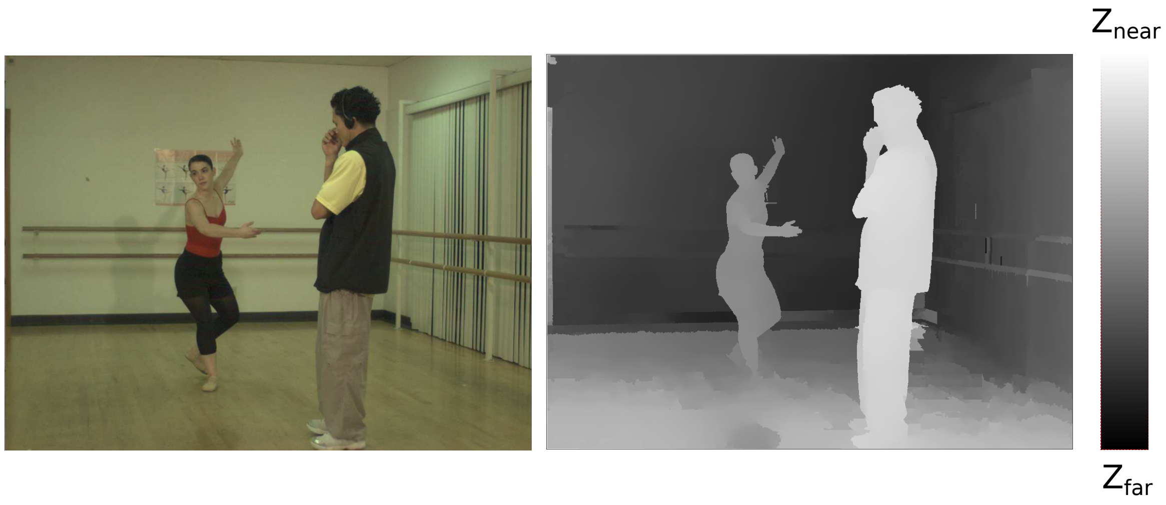


Figure 2.2: This image shows a frame of a 3D video and its corresponding depth map. Depth map is a gray-level image which shows the depth of each pixel

Various kinds of display systems are developed for 3D videos. Some 3D displays need special kind of glasses like polarized glasses. Polarized glasses use orthogonal or circular polarizing filters for separating left and right views which are projected superimposed onto the same screen. A more recent group of 3D displays do not need any glasses. These displays, called auto-steroscopic, use a lenticular sheet or parallax barrier element in front of the light emitters to ensure correct view separation for different viewing angles [7].

## 2.1.2 Interest Points of Video Frames

In our proposed signature for the texture signal of 3D videos, we make use of interest points of video frames. Interest points or local features are computer vision terminology which are defined as: "A local feature is an image pattern which differs from its immediate neighborhood. It is usually associated with a change of an image property or several properties simultaneously, although it is not necessarily localized exactly on this change. The image properties commonly considered are intensity, color, and texture." [42]. Interest points are often described by the appearance of patches of pixels surrounding the point location. As an example, in Figure 2.1.2, patches (a) and (b) with large contrast changes (gradients) convey more information, and are more likely to be found and matched in a similar image

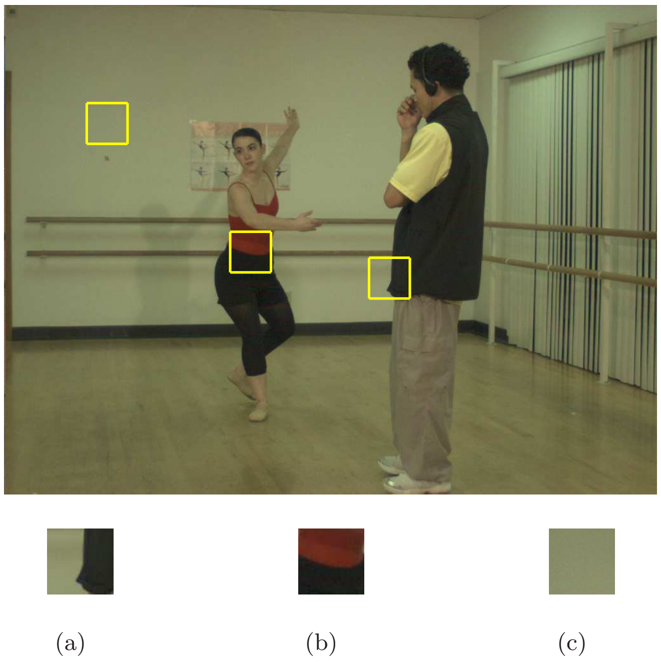


Figure 2.3: Patches (a) and (b) can be localized or matched with higher accuracy than (c).

than textureless patch (c).

Interest point extraction has two phases. The first phase is to search an image for locations that are likely to match well in other images. The second phase is to describe the region around detected interest points in a compact and stable form that can be matched against other descriptors [39]. Many interest point extractor algorithms have been proposed in the literature. The important key of a good interest point extractor is repeatability: Same features should be found in two images of the same scene taken under different viewing conditions. Mikolajczyk et. al. [28] compare the performance of some descriptors on real images with different geometric and photometric transformations including rotation, scale change, viewpoint change, image blur, JPEG compression, and illumination. The results show SIFT and SIFT-based methods obtain the best results.

Scale Invariant Feature Transform (SIFT) [24] detects potential interest points using extrema of difference of Gaussian function, then each interest point is described around its neighborhood by a gradient orientation histogram, and the bins are stored in a 128-dimensional vector. SIFT method is proposed in 2004. Many methods have been developed since then that try to improve its performance. Speeded-up robust features (SURF) [25] is one of them which is much faster. As we will explain in Section 3.2.4, we exploit SURF interest points because of its high performance and fast computation.

### 2.1.3 Video Copy Detection

Video copy detection is important for copyright infringement issues. In order to study this problem more fully, we must first decide what is meant by a copy. A copy defines as a set of tolerated transformations from the original content. A tolerated transformation is one which when applied to the original video, produces a transformed video which is still recognizable as the original video. video transformations, can be done intentionally to avoid detection or unintentionally because of the copying process. For example, a copied 2D video may be scaled, rotated, cropped, transcoded to a lower bit rate, or embedded into another video. The contrast, brightness, or colors of video can also be changed. In addition to mentioned spatial transformations, temporal transformations may also be applied to copied videos. Fast or slow motion, random replication or omission of frames, and changes in frame rate due to different video standards are typical temporal transformations.

Detecting copies of 3D videos is even more challenging. This is because 3D videos have many more transformations than 2D videos. First, each 3D video has at least two views, where each view is a 2D video. 2D traditional transformations can be applied on one, all, or subset of the views, resulting in many possibilities for transformations. Second, 3D videos can be encoded using different formats, including stereo, multiview, video plus depth, and multiview plus depth. Changing from one format to another complicates the detection process. For 3D formats that have depth signal, several transformations can be applied on the depth as well, such as depth blurring. Furthermore, a copied 3D video can contain a subset of the views in the original video. Finally, new views can be created (synthesized) using techniques such as the one in [35] and [30]. Synthesized views display the scene from different angles, and thus reveal different information than in the original views. For example, an object occluded in one view could appear in another.

There are different methods for video copy detection. One method called watermarking [16] is to embed information which is both distinctive and invisible to the user into the content. Then, copy detection becomes a matter of searching the video content for this hidden information. Another method is to use content itself. This is known as Content-Based Copy Detection (CBCD). The underlying premise of content-based copy detection is that the content itself is the watermark. In other words, there is enough information in the content to create a unique fingerprint of the video. These kind of methods involve extracting the fingerprint from the content and performing a distance measure to determining the similarity between the fingerprints of the query video and the original videos.

## 2.2 Related Work

As mentioned in the previous section, watermarking and content based video copy detection are the two approaches to find copies of videos. 3D watermarking approaches in the literature can be classified into three groups [37]: (i) 3D/3D: Watermark is embedded in the 3D model, and it is detected in the 3D model; (ii) 3D/2D: Watermark is embedded in the 3D model, and it is detected in the 2D rendering; and (iii) 2D/2D: Watermark is embedded in the 2D rendering, and it is detected in the 2D rendering. The first two groups try to protect the traditional representation of a 3D scene, which are geometry and texture. The third group tries to watermark the sequences of images which are the 2D projections of the same 3D scene. Consequently, the third group can be used for copy detection of 3D videos [37]. While the first two groups have been studied quite widely, the third group emerged after image based rendering techniques developed [20].

Alper et al. [20] propose a watermarking scheme for multiview 3D videos. They embed the watermark into the main representation of a multiview 3D content and extract it after the content is transformed or a virtual view is generated. Their research is limited to static scenes consisting of one object or one depth layer. Also, this watermarking scheme only considers multiview 3D format, not depth enhanced formats.

The content-based copy detection of 3D videos is fairly new problem. The only work that we are aware of is by Ramachandra et al. [33] where they propose a method to protect multiview 3D videos using a fingerprint based on scale invariant feature transform (SIFT) [24], a local feature extractor and descriptor. They extract SIFT descriptors of each of the views of a multiview query video, and compare it to those of an original video. A problem with this work is that their evaluation is performed at the frame level, and the authors do not explain how they decide whether a video is a copy or not, nor do they identify the location of a copied clip in the reference video.

Although 3D copy detection methods are scarce in the literature, there are many methods available for 2D video copy detection. Hampapur et al. [13] use the temporal features of the video as the fingerprint. They describe a signature based on motion of the video across frames. In a similar way, Tasdemir et al. [40] use the motion vectors for the signature of a frame. Some other methods [21] [46] use fingerprints which are obtained directly from compressed videos. Another group of methods use color histograms as videos' fingerprint. For instance, [13] uses YUV color space. It quantizes Y into 32 bins and each of U and V into 16 bins to produce a 64 bin histogram. The color histogram signature is the sequence of histograms at each frame. Matching is done by maximizing the histogram intersections between the test and the reference video. The color histogram signature is prone to global variations in color which are common when recoding video. Other group of methods use interest points of video frames as signature. Liu et al. [23] use local features that are extracted by SIFT as the frame signature. Roth et al. [34] take every frame of the source video and divide it into 16 regions. They then use Speeded-Up Robust Features (SURF) [25] to find local points of interest in the frame.

Other than these methods which exist in this research area, there are some companies which actually deployed content based video copy detection to find commercial videos copies. Audible Magic[1], Irdeto[2], Civolution[3], and Vobile[4] are some of the well-known companies offering video copy detection and other video services which use fingerprinting technology such as broadcast monitoring and content aware advertisement. Of course, they do not reveal their methods, but all of them mention the use of videos' content signature. They also exploit file's name, file's size, source IP address history, previous transactions, metadata, watermark, and audio signature to make robust decisions.

Although all of these methods can be used for 3D video copy detection, they are designed for 2D videos, and they ignore the information in different views and the depth of videos, which are important especially in the presence of 3D video transformations such as view synthesis. The importance of using depth and visual information together to increase the performance is shown in Section 5.9.

[1]http://www.audiblemagic.com/

[2]http://irdeto.com/

[3]http://www.civolution.com/home/

[4]http://www.vobileinc.com

# Chapter 3

# Proposed 3D Video Copy Detection System

In this chapter, we start by presenting an overview of the proposed system and how it can be used. Then, we present the details of its different components. Then, we analyze its space and time complexity.

## 3.1 Overview

We propose a novel system to detect 3D video copies. Figure 3.1 shows a high-level illustration of the system. The system can be used in different scenarios, including the following:

- Content Owners. A copyright owner can deploy the copy detection system, which periodically crawls online sites and downloads recently posted videos. These videos are then compared against the owners' videos to find potential copies.
- Hosting Sites. A video hosting site can offer content owners a service of detecting copies of their copyrighted materials for possible actions. Copies of certain videos could even be prevented from being posted on the hosting web site.
- Third-party Offering Video Copy Detection Services. A third-party company can deploy the video copy detection system on behalf of one or more content owners.

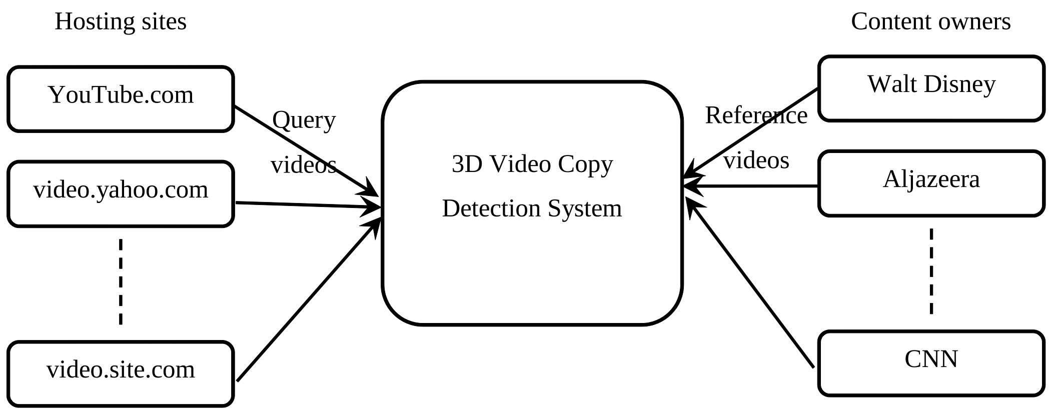


Figure 3.1: High-level illustration of the video copy detection system.

The 3D video copy detection system has two main components: Processing Reference Videos and Comparing Query Videos. We describe each of them in the following. Note that, we refer to original videos as *reference* videos. Videos that we check against reference videos are called *query* videos. If a query video matches one of the reference videos, that query video is called a copied video.

## 3.2 Processing Reference Videos

The first component of the system is Processing Reference Videos, summarized in Figure 3.2. Each reference video is processed once to create its signature, which is later used to detect potential copies. The signature is composed of depth information as well as visual features extracted from frames of the video. Signatures of reference videos are stored in a way that facilitates searching and comparison against query videos. Description of each component is presented below.

### 3.2.1 Extract Depth

For 3D videos encoded in video plus depth format (or its variants), the depth information of each frame is usually represented as a gray-level image showing the depth of each pixel in that video frame.

For 3D videos encoded in stereo or multi-view formats, where no depth information is explicitly given, a method for estimating the depth information is used, which is based on

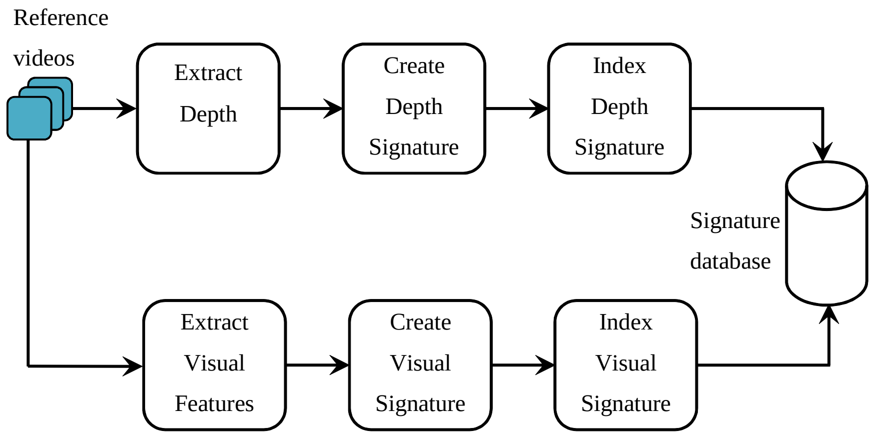


Figure 3.2: Steps for processing reference videos

the following. Human eyes are horizontally separated by about 50-75 mm depending on each individual. Consequently, each eye has a slightly different view of the world. This difference between the points of projection in the two eyes is called binocular disparity. Disparity between a stereo pair of images can be used to extract the depth information, since the amount of disparity is inversely proportional to the distance from the observer. Generating disparity images is called stereo matching, which is the process of taking two or more images and estimating a 3D model of the scene by finding corresponding pixels in the images and converting their 2D positions into 3D depths. Szeliski et al. [38] provide taxonomy of methods available in the literature for correspondence matching. It is worth mentioning that there are both hardware-based and software-based approaches available to generate depth information in real-time [44].

### 3.2.2 Create Depth Signature

After extracting the depth map which is a gray-level image, the depth signature is computed in two steps. First, as shown in Figure 3.3, the depth map is divided into a grid. The division can be uniform, i.e., into equal size blocks, or non-uniform to account for different importance of the regions in the depth map. The number of blocks in the grid is a configurable parameter which trades off the computational complexity with the copy detection accuracy. We found out that a $20 \times 20$ uniform grid give us a good accuracy in an acceptable time.

In the second step of creating the depth signature, the blocks of the depth map grid are

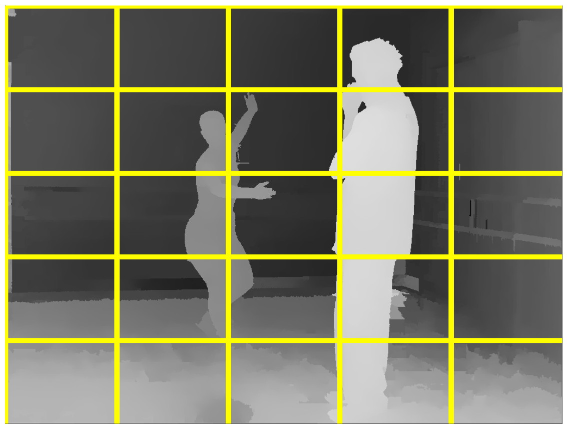


Figure 3.3: In the first step of the depth signature extraction, the depth map is divided into a grid.

summarized into a vector, where each element of the vector represents one block. Various metrics can be used to summarize the depth information in each block. We use mean of the depth values in each block to summarize the whole block. More complex metrics that are composed of multiple components, e.g., the mean and standard deviation, can also be used. The depth signature for a video frame takes the form $< d_1, d_2, ..., d_D >$, where D is the total number of blocks in the depth map grid, and $d_i$ is the mean of depth values in block $i$.

The depth signature can be created for every frame in the video. It can also be created for only a subset of the frames in order to reduce the computational complexity. This subset of the frames can be chosen deterministically, e.g., each 10th frame is chosen, or randomly. In addition, the subset of the frames can be the keyframes of the video, where a keyframe is a representative frame for a sequence of video frames containing similar visual information, which is referred to as a video shot. Shot boundary detection algorithms such as [22] can be employed to identify when a shot starts and ends. Keyframe selection algorithms such as [11] can be used to select key frames. The depth signature of a video is composed of the depth signatures of its frames, or the chosen subset of frames for which the depth signatures are created.

### 3.2.3 Index Depth Signature

Depth signatures are vectors with multiple dimensions. These vectors will need to be compared against depth vectors from other videos in order to find potential copies. We index depth signatures in order to facilitate these comparisons. In particular, given a depth vector from a query video, we are interested in finding the closest depth vectors from the reference video database. Multiple methods such as randomized kd-tree, k-means, and locality sensitive hashing (LSH) can be used to achieve this nearest neighbor search efficiently. Based on the requirements of the system, like the performance or the need to be dynamic, a method can be chosen. We propose to use kd-tree for nearest neighbor search for large-scale video copy detection system, because it is dynamic and can be distributed for scaling purposes. More explanation is provided in Section 4.3 for this choice.

Here we present a brief background on randomized kd-tree to find approximate nearest neighbors, and then show how it can be used in our system. In the standard version of kd-tree [10], starting with $N$ points in $R^d$, the data space is split on dimension $i$ in which the data exhibits the greatest variance. An internal node is created to store dimension $i$ and its median value, so an equal number of points fall to left and right subtrees. Then, the same process is repeated with both halves of the data, and finally the data are stored in the leaves of the tree. Since median values are used, the created tree is balanced with depth $\lceil log_2 N \rceil$. To find the nearest neighbor to a query point $q$, first the leaf that point falls in is found. The point in this leaf is a good candidate for query point nearest neighbor. Then a backtracking stage begins in which whole branches of the tree can be pruned if the region of space they represent is further from the query point than the closest neighbor seen so far. The search terminates when all unexplored branches have been pruned. However, this approach is not efficient in high dimensional spaces, and also we are looking for approximate nearest neighbors, not the exact one. So we can limit the number of leaf nodes we are willing to examine, and return the best neighbors found up to that point.

Silpa-Ann and Hartley [36] proposed an improved version of kd-tree in which multiple randomized kd-trees are created, and simultaneous searches are performed using several trees. Randomized kd-trees are created by choosing the split dimension randomly from the first $D$ dimensions on which data has the greatest variance. They also use a priority queue across all trees to explore nodes based on their distances to the query, instead of backtracking based on the tree structure. Their experiments show that this technique leads

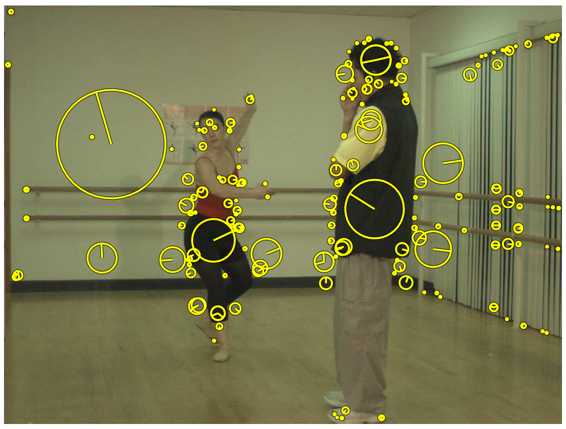


Figure 3.4: Visual features of a video frame which will be used to compute the visual signature.

to 3-times search speedup with the same performance. We used this improved version in our system, and for each signature we store the following fields: $< VideoID, FrameID, ViewID, DepthSignature >$, so that whenever a signature is returned, we know its corresponding video, frame, and view.

### 3.2.4 Extract Visual Features

We extract features from individual frames of videos. Visual features should be robust to, i.e., do not change because of, various transformations such as scaling, rotation, change in viewpoint, and change in illumination. Different types of visual features can be used in our system, including but not limited to SURF [25] and SIFT [24]. In our implementation, we use SURF features, which outperform previous methods with respect to repeatability, distinctiveness, and robustness, and also can be computed much faster [25]. Figure 3.4 shows an example of the visual features of a video frame.

### 3.2.5 Create Visual Signature

In our previous work [18], we used actual feature descriptors, SIFT descriptors, as visual signatures. This visual signature is large and hard to compare, since there are 200 signatures and each has 128 elements per frame. Using the number of visual features instead of their descriptors has been shown to provide good performance in [34] and [14]. Thus, in the

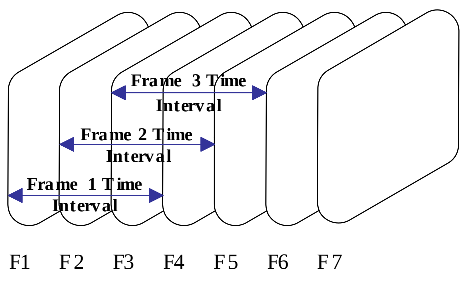


Figure 3.5: Frames considered in visual signature computation of each frame.

proposed system, we use a modified version of the signature introduced in [14] where only the count of visual features is used. This visual signature is spatio-temporal. Spatial information is obtained by dividing each video frame into regions. In each region, local SURF features are found and the number of features in each region is counted. To add the temporal information to the signature, the counts in each region are sorted along a time line and an ordinal value based on its temporal rank is assigned. We note that in [14], one signature is computed for the whole query video, which makes detecting copied videos embedded in other videos difficult. To overcome this, our proposed signature generates a spatial-temporal signature for every frame, but we include a limited temporal information in each one. As shown in Figure 3.5, we generate the signature of a specific frame by considering a time interval that starts at that frame, and has a specific length which defines how much temporal information we want to include in each signature. For example, in our implementation, we used a 10-frame interval, and a $4 \times 4$ grid, which results in a $4 \times 4 \times 10 = 160$ element vector signature per frame. Compared to the old visual signature, it is much smaller, but as will be shown in the experiments section, is still distinctive and gives good results.

An example of extracting this signature is shown in Figure 3.6. To make it simpler to demonstrate, a $2 \times 2$ grid and a four frame wide interval are considered for signature extraction. In Figure 3.6(a), each frame is divided to four regions, and the feature count in each region is shown inside it. To extract each frame signature, the ordinal scores of each region in the frame time interval is computed. The ordinal scores of the first region, which is shown in gray background, are shown above the frames for frames 1 and 2. In Figure 3.6(b), the same procedure is performed for other regions, and finally the frame signature is the ordinal scores.

The visual signature for a video frame takes the form $< v_1, v_2, ..., v_V >$, where $V$ is

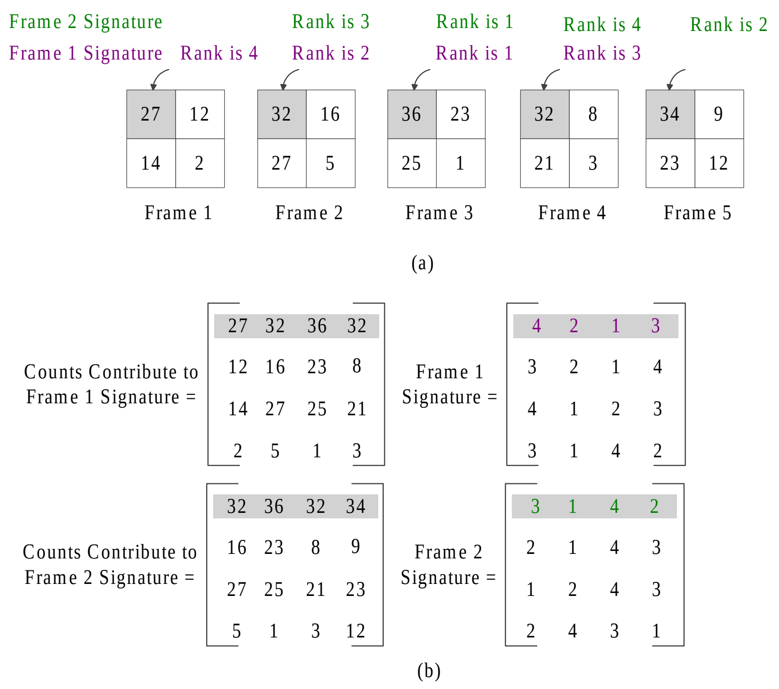


Figure 3.6: An example of computing the visual signature.

Number of Blocks × Number of Frames in the time interval, and $v_i$s are ordinal scores.

Similar to the depth signature, the visual signature can be created for every frame in the video, or a subset of the frames in order to reduce the computational complexity. The visual signature of a video is composed of the visual signatures of its frames, or the chosen subset of frames for which the visual signatures are computed.

### 3.2.6 Index Visual Signature

Visual signatures are vectors with multiple dimensions. These vectors will need to be compared against visual vectors from other videos in order to find potential copies. Like depth signatures, we index visual signatures in order to facilitate these comparisons. Again, we use kd-tree to index visual signature, and the points added with the following fields: $< VideoID, FrameID, ViewID, VisualSignature >$, so that whenever a signature is returned, we know its corresponding video, frame, and view.

## 3.3 Processing Query Videos

The second component of the proposed system is Comparing Query Videos, summarized in Figure 3.7. The depth signature is first computed from the query video. As mentioned before, the signature can be computed for only a subset of video frames. Subsampling is more important in case of query video, since we want to respond to a query as fast as possible. The methods used to extract query signatures are the same as the ones used to process reference videos. Then, the depth and visual signature of the query video is compared against the depth and visual signatures in the reference video database. If there is no match, the query video is not considered for any further processing. If there is a match, a combined score is computed based on the depth signature and visual signature matching scores. Finally, the combined score is used to decide whether the query video is a copy of one of the videos in the reference video database. If a query video is found to be a potential copy of a reference video or part of it, the location of the copied part in the reference video is identified. More details are provided in the following.

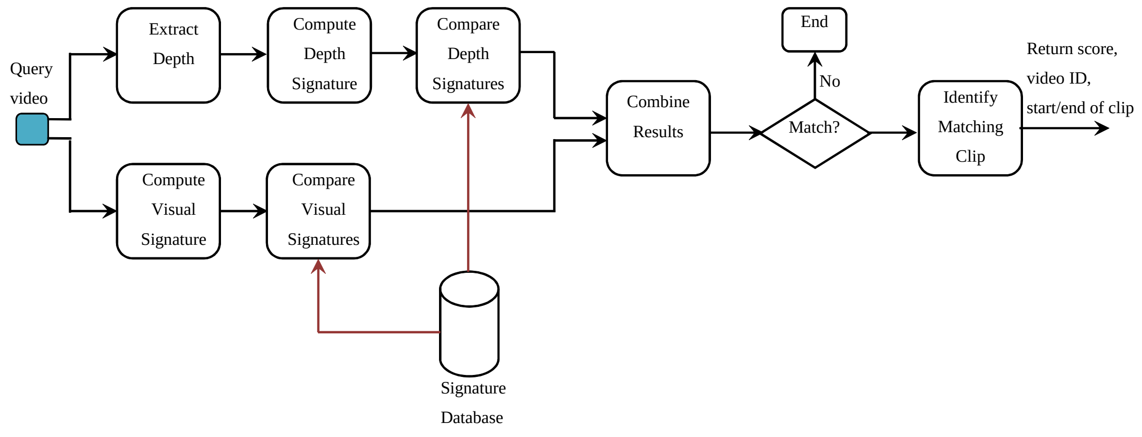


Figure 3.7: Comparing query video against reference videos.

### 3.3.1 Compare Depth Signatures

Finding the potential copied videos using depth signature takes place in two main steps: frame level comparison and video level comparison. The goal of the first step is to find the best matching frames in the entire database for each query frame and compute a score between each matched pair. The goal of the second step is to account for the temporal aspects of the video frames, and to compute a matching score between the query video and each reference video.

In the first step, for each depth signature of the query video, the depth signatures that are closest to it based on their Euclidean distance are found using the nearest neighbor search method. Using kd-tree, for each depth signature a fixed number of its approximate nearest neighbors and their distances are found. Distances can be used as matching scores. Alternatively, a threshold can be used such that scores for distances exceeding the threshold can be set to zero and other scores are set to 1. This will reduce the computation needed to compare scores. It should be noticed that the frames found in this may belong to different videos.

In addition, 3D videos can have multiple views, and a signature from the query frame should be checked against frames from different views. Two frames are considered a match if at least one of their views matches. Finally, a score is computed for each matched pair of frames using the distance of their views. At the end of this step, the number of matched frames in each reference video is counted. Then, reference videos with the number of matched frames exceeding a threshold are considered in the next step. Other videos are no longer considered.

In the second step of the depth signature matching, the temporal characteristics of the videos are considered. Temporal characteristics mean the timing and order of the frames in the query and reference videos. For example, if frame $x$ in the query video matches frame $y$ in the reference video, we expect frame $x + 1$ in the query video matches frame $y + 1$ in the reference video. This is important to account for as copied videos are typically clips with contiguous frames taken from reference videos. Also, a copied video can be embedded in other videos.

In order to consider the temporal characteristics, a matching matrix is computed for each candidate reference video and the query video. The columns of this matrix represent reference video frames, and the rows represent the query video frames. Entries in the matrix are the relevance scores. Figure 3.8 shows an example, where dark squares represent matched frames. Using this matrix, the longest sequence with the largest number of matching frames is considered as a potential copy. If the frame rate of the original video and its copy are the same, this sequence's gradient (slope) would be equal to 1. However, since change of frame rate transformation may be applied to the copied video, we also consider sequences with gradients 0.5, and 2. These gradients correspond to two extreme cases when the frame rate of the query video is changed to half and twice of its original, respectively. Other frame rate changes are between these three cases, and are still detected, as it is examined in Section 5.7.

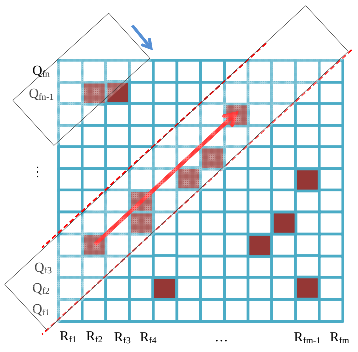


Figure 3.8: Matching matrix for frames from query and reference videos considering their timing.

It is worth mentioning that frame dropping and occasional frame mismatches caused by possible transformations must be taken into account. Thus, the sequences mentioned before are not strictly linear and gaps may exist. So, to find the longest sequence with the greatest score, instead of considering a line of frames, a band with a specific width is considered, as shown in Figure 3.8. This band, with one of the gradients mentioned above, starts sweeping the matrix from top left most position and moves one block each time. At each position, the temporal score of the longest sequence of matched frames inside the band is computed. After performing this process for all positions, the position with the greatest temporal score is considered the potential copied location, and its score is considered the depth matching temporal score of the reference video.

### 3.3.2 Compare Visual Signatures

Like depth signature comparison, visual signatures comparison takes place in two steps, frame level, and video level. First, for each query frame visual signature, the visual signatures that are closest to it based on their Euclidean distance are detected using nearest neighbor search method. Using kd-tree, for each visual signature a fixed number of its approximate nearest neighbors and their distances are found. These returned signatures may belong to different videos. To find the matched videos, the number of matched frames in each reference video is counted, and videos with the number of matched frames exceeding a threshold are considered a match. At the video level comparison, like the one explained for depth video level matching, the temporal characteristics are taken into account, and a temporal score is computed between the query video and each potential reference video. Finally, the best matching videos based on their temporal scores are considered as potential copies.

### 3.3.3 Identify Matching Clip

Copied videos can be small clips of the reference videos. It is useful to automatically identify the location of a copied clip in the reference video. We use the matching matrix shown in Figure 3.8 to identify the location of the copied clip. Notice that we have two matching matrices: one from matching depth signatures and the other from matching visual signatures. We can either use one of them or both. We find the longest diagonal sequence with the greatest score in each case. The start and end of the longest sequence give the start and end location of the copied clip in the reference video. Using both of depth and visual matching matrices can yield more accurate locations of the copied clips. In this case, the intersection of the two sequences returned from comparing depth and visual signatures is used to mark the start and end location of the reference video.

## 3.4 Algorithm Analysis

We analyze the space and time complexity of the proposed system. Space complexity refers to the storage needed to store the signatures of reference videos. Signatures of query videos are created online and compared against the signatures of reference videos in the database. We analyze the space complexity as a function of total number of frames in all reference videos. We use kd-tree to index signatures in order to facilitate nearest neighbor search. To calculate the space needed to store the index, we note that we have to store signatures, their information, and the trees. To store the signatures themselves, in addition to signatures' feature vectors, we have to store video ID and frame ID of each signature. We assume that we use 4 bytes for video ID, and 4 bytes for frame ID of each signature, which is large enough for long videos and large video databases. To store the trees, we have to store internal nodes and leaves. In a kd-tree with $n$ leaves, there are $n-1$ internal nodes. The internal nodes need to store their splitting dimension and the threshold values (assumed to store each of them in a single byte), and the leaves need to store the index of the signature, which can be stored in $log_2 n/8$ bytes. Thus, the space required to store a kd-tree index can be computed using Equations 3.1:

$$\begin{aligned} Storage &= n(sd+4+4) && \text{Storing signatures and their information} \\ &+ n(2T_{kd}+T_{kd}\tfrac{log_2 n}{8}) && \text{Storing the trees} \end{aligned} \tag{3.1}$$

where $d$ is the signature dimension, $s$ is number of bytes per feature, and $T_{kd}$ is the number of kd-trees. Now we replace the general terms with the specific values of our system. Let the total number of frames in all reference videos be $N_r$. Also, in our implementation we used FLANN library, with 4 trees, so $T_{kd} = 4$. The dimensions of depth signature, and visual signature are 400 and 160, respectively. Each depth signature feature is between 0 and 255, and each visual signature feature is between 0 and 9, so for both features we have $s = 1$ byte. Using these parameters, the storage required to store our depth and visual indexes is $O(N_r log_2 N_r)$.

Next, we analyze the time complexity, which is the time needed to process a given query video of length $N_q$. In our analysis, we consider the worst case scenario, which happens when the query video matches one of the reference videos. We assume that the time taken to compute the depth and visual signatures for a frame is $T_d$ and $T_v$, respectively. $T_d$ and $T_v$ do not depend on $N_q$, but depend on the frame resolution, which is constant. Comparing signatures and identifying location of copied clip do depend on $N_q$.

For comparison of signatures, we search through the kd-tree to find approximate $m$ nearest neighbors. This involves scaler comparison operations at each level of the tree, limited number of backtracking (typically $B \sim 250$) to examine more leaves, computing the distance between query and each leaf signatures, and linear search in the final list of signatures $B$ to find the best $m$ matches in the list. We know that the height of the tree is $log_2 N_r$, and that computing the distance between query and leaf signatures can be performed with $d$ multiplications and $d$ additions. So the running time of a single search in the tree is $B(2d + log_2 N_r) + B$. Thus, for comparing the depth and visual signatures for a query video that has $N_q$ frames, we need $250N_q(801 + log_2 N_r)$ and $250N_q(321 + log_2 N_r)$ time, respectively.

In the Spider system, we found out that using $m = 50$ suffices for good performance. So the $m = 50$ best matching frames for each frame are used to vote for corresponding reference videos, and $K$ best matching videos are found in $mN_q + R$, where $R$ is the number of reference videos in the database. Then, we compare these $K$ reference videos against the query video, constructing the matching matrix, such as the one shown in Figure 3.8, between the query video and the reference video, which takes $LKN_q$ steps, where $L$ is a constant referring to the number of frames in the longest video. Finally the temporal score is computed using this matrix. To do so, as explained in Section 3.3.1, each matrix element is traversed once, and this takes $LN_q$ addition, at most, to compute the temporal score and matched location. Adding all up, the worst case running time to process a query video of length $N_q$ is:

$$\begin{aligned} Time &= 250N_q(801 + log_2^{N_r}) + 250N_q(321 + log_2^{N_r}) + 2(50N_q + R + LKN_q + LN_q) \\ &= 2R + N_q(280600 + LK + L) + 500N_q log_2 N_r \\ &= O(N_q log_2 N_r) \end{aligned} \tag{3.2}$$

The above equation considers the fact that $L$, $K$, and $R$ are constant and much smaller than $N_r$.

# Chapter 4

# Implementation

[1]We implemented all steps of the proposed 3D video copy detection system, which we call Spider. We modified and integrated several open-source libraries in our system. We provide some highlights of our implementation in the following.

## 4.1 Overview

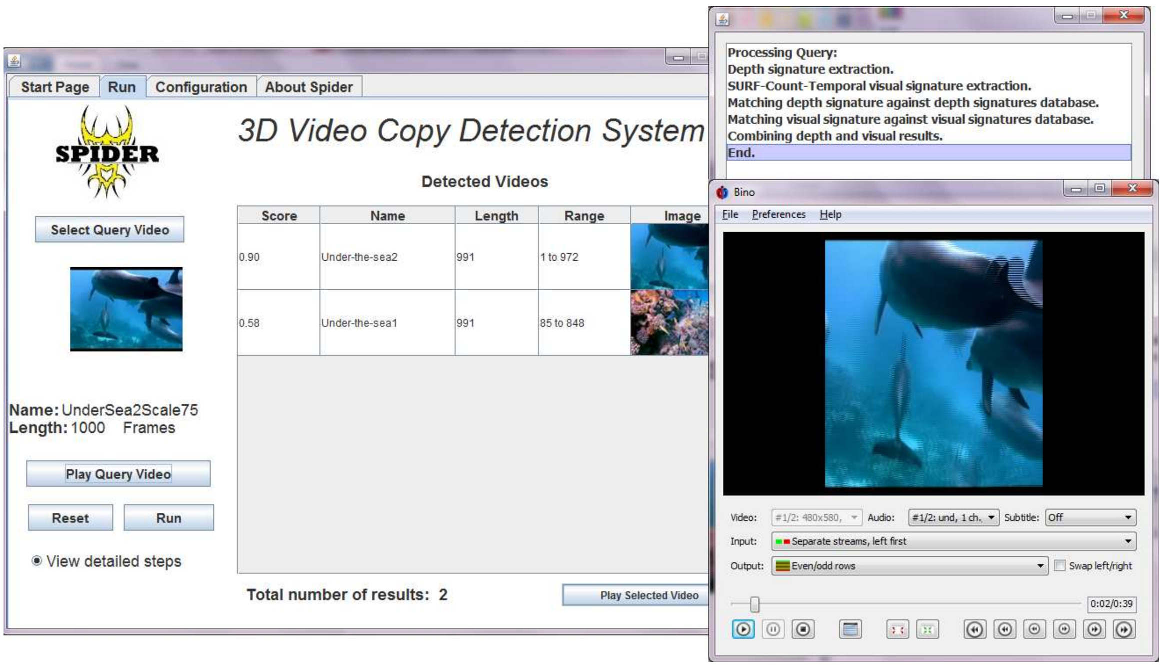
We implemented Spider, including its graphical user interface, in Java. Figure 4.1 shows a snapshot of Spider system. The user provides the location of the reference videos that need to be protected, and then the location of a query video that wants to match it against the reference videos. The Spider system compares the selected query video with the database, and returns the detected videos, if any, in a table sorted according to their scores.

The main components of the Spider system, Core and UI, are shown in Figure 4.2. The Core package is the implementation of the main copy detection algorithm, and UI provides a graphical user interface for users to work with the Core. We explain the most important components in the following.

## 4.2 Detection Engine

DetectionEnging is the main class that controls everything from signature extraction to matching. It has close interaction with the Video class. Each video in the system is an

[1]A. Abdelsadek contributed to the implementation of the system, especially FLANN library integration and data collection and processing.



Name: UnderSea2Scale75
Length: 1000 Frames

| Score | Name | Length | Range | Image |
|---|---|---|---|---|
| 0.90 | Under-the-sea2 | 991 | 1 to 972 | |
| 0.58 | Under-the-sea1 | 991 | 85 to 848 | |

Total number of results: 2

Processing Query:
Depth signature extraction.
SURF-Count-Temporal visual signature extraction.
Matching depth signature against depth signatures database.
Matching visual signature against visual signatures database.
Combining depth and visual results.
End.

Video: #1/2: 480x580, Audio: #1/2: und, 1 ch. Subtitle: Off
Input: Separate streams, left first
Output: Even/odd rows
Swap left/right
0:02/0:39

Figure 4.1: A screen shot of the demo software. The query video, shown on left, which is scaled version of a reference video is detected to be a copy of the reference videos shown in the table.

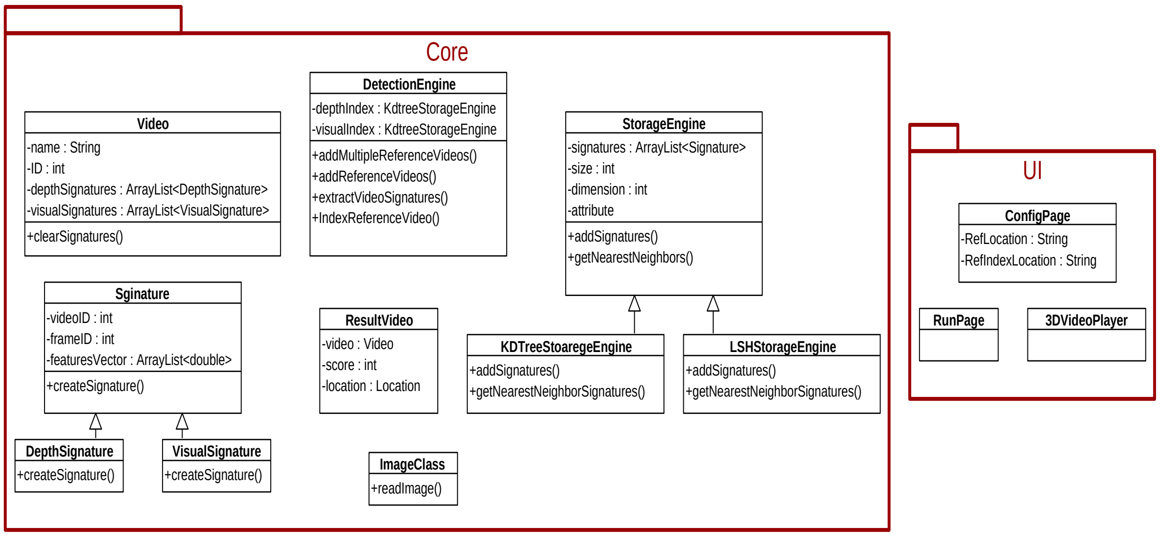


Figure 4.2: Main components of the demo software.

object that has its own signatures, depthSignatures and visualSignatures, as fields. These two fields are vectors of Signature instances, where each element of these vectors are the signature of one video frame. In the first run of the Spider, the user provides the location of the reference videos that needs to be protected. DetectionEnginge extracts their signature, indexes them, and stores the indexes on the disk, so it can load them in the next runs, instead of repeating this step in each run of the software. The user can also add other reference videos to the index later.

In processing reference videos phase, DetectionEngine first extracts video frames, texture and depth, from videos, which have different formats. It uses the FFmpeg [8] package to extract the frames in grayscale format, since the depth is grayscale itself, and SURF works with grayscale images, and it takes less space to save them as well. We use "ffmpeg -i video frame_%4d.pgm" command to extract the frames, distinguish them with 4 digit numbers, and store them in pgm grayscale format.

Then, these frames are read to the memory for processing. The depth signature of each frame is computed in two steps. First, the depth frame is divided into a 20x20 equal-size grid. Then, the average of the depth values (pixels' intensity) in each grid is computed and stored in order in a 400 dimensional vector field called featuresVector in an instance of DepthSignature class. In the second step, the depth signature is indexed to facilitate nearest neighbor search needed in the query processing phase. More explanation on indexing and storage is provided in Section 4.3. For the visual signature, the system extracts SURF features using jopensurf [15] open-source library, which is implemented in Java. Then, each texture frame is divided into a $4 \times 4$ grid, and the number of visual features in each block of the grid is counted. Then, the counts of the frames in an interval of 10 frames are sorted, and an ordinal score is assigned to each block of each frame. These ordinal scores are stored in featuresVector field of a VisualSignature instance. The details of extracting visual signatures is explained in section 3.2.5. After extracting the visual signatures, they are indexed as well. As mentioned before, the indexes are stored on disk for later use.

In the query processing phase, a query video is selected by the user. Then, the software extracts its frames and signatures, similar to steps taken for reference videos. To match the query video signatures against reference videos signature, the indexes are used to find the best matches or nearest neighbors. Section 4.3 provides more details on nearest neighbor search in the indexes. Finally, these matched signatures are used to find the best matching videos as described in section 3.3. The matching takes place both for depth and texture of the query. Each matching step returns the reference videos that are detected to be the original version of the query video, and a score between $[0, 1]$ is assigned to them. Then, the scores of these two matching steps are combined. For each reference video returned, its score will be the weighted sum of their scores. We used the same weight of 0.5 for both depth and texture matching scores. The returned reference videos are shown as the result to the user.

## 4.3 Index Storage

Multiple methods such as randomized kd-tree, k-means, and locality sensitive hashing (LSH) can be used for indexing to achieve approximate nearest neighbor search efficiently. Based on the requirements of the system, like the performance or the need to be dynamic, a method can be chosen. Our first choice was LSH [6] for nearest neighbor search for large-scale video copy detection system, because it can be easily parallelized. However, we should consider the fact that our data is very dynamic. In other words, videos may be added to the index at any time, and any newly added video will add many points to the index. We have two choices to deal with this situation in LSH. First one is to keep the table sizes fixed, which will put more and more points in the same bucket, and reduce the accuracy. Second one is to resize the hash tables periodically, and rehash all the points again, which will reduce the performance. Thus, number of hash functions or bin sizes parameters affect the trade off between run time and accuracy, and they have to be tuned for every database size under consideration, and we should not use the same settings when enlarging the database. This made LSH less appealing for our software, since the reference videos will be available gradually.

Our next choice was randomized kd-tree [Silpa-Anan and Hartley 2008], since it is dynamic and has good performance. According to [2] which did a thorough comparison between different indexing approaches for high dimensional data, kd-trees provide the best overall trade off between performance and computational cost. In their experiments, they observed that its run time grows very slowly with the number of points while giving excellent performance. Moreover, with smart parallelization schemes, like the one proposed and implemented in [1], it can significantly speedup when running on multiple machines, so it can be scaled to large video databases. The only drawback of kd-trees is larger storage requirements, which can be addressed by using more compact signatures.

In the Spider system, we use the randomized kd-tree implementation of FLANN library version 1.6.11 [9] which uses five dimensions to build the tree, and we use 4 randomized kd-trees. FLANN library implementation details and choices can be found in [32].

## 4.4 Input/Output Management

As mentioned earlier, the user provides the the location of the reference and query videos via the ConfigPage of the graphical user interface. The system uses the videos' location to extract and store the frames in a specific location. Then, these frames are read to the memory, one by one, by ImageClass as buffered images to speed the reading process.

The output of the system are instances of ResultVideo class. As shown in Figure 4.1, the information provided to the user are the name, score, length, and a sample image of the detected reference videos. It also shows the location, start and end frames, in the reference video which is detected to be the copied portion of the query video. The user can also play the query and reference videos to compare them visually. When the user wants to play the videos, the system launches the Bino [3] 3D video player. The Spider system has a verbose option, in which the user can see the detailed steps as the system is performing them.

# Chapter 5

# Experimental Evaluation

We conducted extensive analysis using real 3D videos to assess the Spider system performance. We start by describing how we prepared our video dataset in the following section. Then, we describe our experimental setup. Then, we present the results of our experiments to show the performance of the proposed system.

## 5.1 Preparing 3D Video Dataset

We collected 37 3D videos with different characteristics from three resources: YouTube, Microsoft, and Mobile3DTV project. Two of the videos, Ballet and Break Dancers, have eight views each, which are placed along a 1D arc spanning about 30 degrees from one end to the other. These two videos are in bitmap format and their camera calibration parameters are generated and distributed by the Interactive Visual Group at Microsoft Research [31]. Depth maps are computed using the method described in [47]. The other 15 videos are obtained from the MOBILE3DTV project [29]. These are stereo videos, with only two views. These videos and their depth signals are provided in YUV format, which we converted to bitmap format using the FFmpeg package [8]. The last 20 videos are stereo videos downloaded from YouTube video sharing website. Then, their depth signals are computed using Triaxes DepthGate [41] software.

To create our reference video database, we use one of the eight views (view0) and its associated depth signal from each of the Ballet and Break Dancers videos. And we use the left view and its associated depth from each of the remaining videos except video TU-Berlin. That is, we create 36 reference videos, which are listed in the first 36 rows in Table 5.1.

TU-Berlin video is used in creating queries.

We create many different query videos to capture most realistic scenarios. Specifically, we first create three types of queries as described below and illustrated in Table 5.1:

- Type 1: Query videos are segments or clips of reference videos. These are the first 36 rows in Table 5.1.
- Type 2: Query videos are segments of reference videos embedded in other videos. These are rows 37 to 42 in Table 5.1.
- Type 3: Query videos contain no parts of the reference videos. These are the last 6 rows in Table 5.1.

Then, we apply different video transformations on the 48 videos shown in Table 5.1. A video transformation means that the video has been modified either intentionally (to avoid detection) or unintentionally (as a result of the copying process). A transformed video is supposed to provide acceptable perceptual quality to viewers, that is, the transformation can be tolerated by viewers. Transformations of 3D videos can be applied on texture, depth, or both. In addition, transformations can be applied individually, i.e., only one transformation is applied on the video, or combined, i.e., multiple transformations are applied on the video at the same time. Texture transformations are chosen from the TRECVID competition transformations. Some of TRECVID[1] transformations like picture in picture, and letterbox are not considered. However, integrating them in the system, using approaches like [14] [23], is straightforward.

We apply 4 transformations on each of the first 20 videos in Table 5.1, which are the stereo videos downloaded from YouTube. These transformations are applied to both views of the stereo videos, and then the depth is extracted from the transformed stereo videos. This results in 80 query videos.

- Video Blurring: Both views are blurred using a radius of 3 for the blur disk.
- Video Scaling: Both views are scaled to 75% of their original size.
- Insertion of Logo: A logo is inserted into both views.

[1]urlhttp://www-nlpir.nist.gov/projects/tv2008/final.cbcd.video.transformations.pdf

Table 5.1: List of videos used in creating query videos.

| | Query | Resolution | nFrames | FrameRate | InsertPoint |
|---|---|---|---|---|---|
| 1 | Boxing | 640×720 | 466 | 30 | 1-466 |
| 2 | Camera-on-Plane1 | 640×720 | 1000 | 24 | 1-1000 |
| 3 | Camera-on-Plane2 | 640×720 | 1000 | 24 | 1-520 |
| 4 | Camera-on-Plane3 | 640×720 | 1000 | 24 | 1-1000 |
| 5 | Car2 | 670×720 | 1000 | 30 | 1-1000 |
| 6 | Cat | 640×312 | 1000 | 30 | 1-1000 |
| 7 | Dimenco-Sample | 960×540 | 572 | 24 | 1-572 |
| 8 | Flowers | 960×1080 | 1000 | 30 | 1-1000 |
| 9 | Football1 | 640×720 | 1000 | 25 | 1-503 |
| 10 | Football2 | 640×720 | 1000 | 25 | 1-1000 |
| 11 | Football3 | 640×720 | 1000 | 25 | 1-1000 |
| 12 | Football4 | 640×720 | 1000 | 25 | 1-1000 |
| 13 | Football5 | 640×720 | 1000 | 25 | 1-1000 |
| 14 | Football6 | 640×720 | 1000 | 25 | 1-1000 |
| 15 | Road1 | 1280×720 | 1000 | 24 | 1-1000 |
| 16 | Road2 | 640×720 | 1000 | 30 | 1-1000 |
| 17 | Skiing | 960×1080 | 1000 | 30 | 1-1000 |
| 18 | Under-the-Sea1 | 640×720 | 1000 | 24 | 1-1000 |
| 19 | Under-the-Sea2 | 640×720 | 1000 | 24 | 1-385 |
| 20 | Under-the-Sea3 | 640×720 | 1000 | 24 | 1-1000 |
| 21 | Alt-moabit | 432×240 | 100 | 24 | 1-100 |
| 22 | Ballet | 1024×768 | 100 | 15 | 1-100 |
| 23 | Book-arrival | 512×384 | 100 | 24 | 1-100 |
| 24 | BreakDancers | 1024×768 | 100 | 15 | 1-100 |
| 25 | Car | 480×270 | 235 | 24 | 1-235 |
| 26 | Caterpillar | 480×270 | 101 | 24 | 1-101 |
| 27 | Door-flowers | 512×384 | 100 | 24 | 1-100 |
| 28 | Flower1 | 480×270 | 152 | 24 | 1-152 |
| 29 | Flower2 | 480×270 | 234 | 24 | 1-234 |
| 30 | Flower3 | 480×270 | 112 | 24 | 1-112 |
| 31 | Grasshopper | 480×270 | 181 | 24 | 1-181 |
| 32 | Hands | 480×270 | 251 | 24 | 1-251 |
| 33 | Horse | 480×270 | 140 | 24 | 1-140 |
| 34 | Leaving-Laptop | 512×384 | 100 | 24 | 1-100 |
| 35 | Rollerblade | 320×240 | 905 | 24 | 1-905 |
| 36 | Snail | 480×270 | 189 | 24 | 1-189 |
| 37 | Berlin-Ballet | Mix | 132 | 15 | 21-81 |
| 38 | Berlin-Break | Mix | 142 | 15 | 31-111 |
| 39 | Berlin-DoorFlowers | Mix | 92 | 25 | 26-86 |
| 40 | Berlin-Flower2 | Mix | 122 | 25 | 11-111 |
| 41 | Berlin-Grasshopper | Mix | 142 | 25 | 11-101 |
| 42 | Berlin-Snail | Mix | 92 | 25 | 11-70 |
| 43 | TU-Berlin1 | 360×288 | 150 | 25 | - |
| 44 | TU-Berlin2 | 360×288 | 149 | 25 | - |
| 45 | TU-Berlin3 | 360×288 | 149 | 25 | - |
| 46 | TU-Berlin4 | 360×288 | 149 | 25 | - |
| 47 | TU-Berlin5 | 360×288 | 149 | 25 | - |
| 48 | TU-Berlin6 | 360×288 | 149 | 25 | - |

- Insertion of Text: Some text is inserted into both views.

We apply the following 9 transformations on each of the last 28 videos in Table 5.1, which results in 252 query videos.

- Video Blurring: This transformation is applied to texture only. The radius of blur disk is chosen randomly from range [0.5 7].
- Video Gamma Correction: This transformation is applied to texture only. The gamma value is chosen randomly from range [0.2 4]
- Video Noise: This transformation is applied to texture only. The noise is chosen randomly from range [0 0.06]
- Crop: This transformation is applied to texture and depth in a way that the same number of pixels are cropped from both texture and its depth. The number of pixels cropped are chosen randomly from range [5 15].
- Logo Insertion: This transformation is applied to texture and depth in a way that the pixels covered by the logo in the video are set to minimum depth.
- Text Insertion: This transformation is applied to texture and depth in a way that the pixels covered by the text in the texture are set to minimum depth.
- Flip: The texture and its depth are flipped.
- Depth Blurring: This transformation is only applied to depth signal. Blurring the depth image smooths sharp horizontal changes in depth images, so the fewer holes would appear in the warped views; however, the warped image quality reduces especially around the not edge areas. The radius of blur disk is chosen randomly from range [0.5 7].
- Depth Noise: This transformation is only applied to depth signal, which adds noise to depth frames and cause quality reduction in the warped images. So the noise cannot be very high, as the quality of the warped image may not be acceptable. The noise deviation is chosen randomly from range [0 0.06]

In addition, we apply two types of combined transformations on these 28 videos: 3 and 5 transformations. For the 3 transformations case, we choose 3 different transformations and apply all of them on each of the 28 videos in Table 5.1. One transformation is applied on the texture, another one the depth, and the third is applied on both. Similarly, for the 5 transformations case, 5 different transformations are applied on each video: two on the texture, two on the depth, and one on both. These combined transformations added 2*28 = 56 videos to our query video database.

We also examine common temporal transformations. Fast or slow motion happen when video's frame rate is changed, while the number of frames is kept fixed. Our system has immunity to this transformation, since it does not change the matching matrix. Random replication or omission of frames is also handled in the matching matrix by considering a band instead of a line of frames. Change of frame rate is the most challenging temporal transformation which keeps the video's total length by dropping, replicating, or merging frames. To test our system against this transformation, FFmpeg [8] package is used to change the frame rate of our 48 query videos. We examine both increase and decrease in frame rate. Table 5.2 shows our approach to generate the transformed versions, which adds 48*2 = 96 videos to our query video dataset.

Table 5.2: Generating query videos with different frame rates.

| Decreased Frame Rate | Original Video Frame Rate | Increased Frame Rate |
|---|---|---|
| 10 | 15 | 24 |
| 15 | 24 | 30 |
| 15 | 25 | 30 |
| 24 | 30 | 50 |

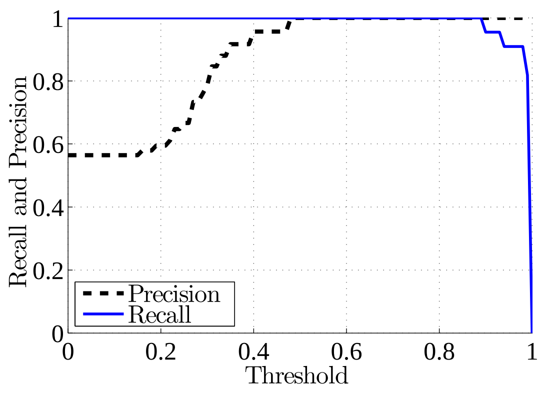
Finally, we apply two *new* transformations which are specific to 3D videos: view synthesis and copying a subset of views. In the view synthesis case, we create additional views from given views using view synthesis tools. Synthesized views present the visual content from different angles to viewers, which could reveal objects that were occluded or present objects with different depths or shades. This means that synthesized views can contain different visual and depth information than the original views. Synthesized views can be created and distributed to enrich users' experience or to evade the detection process. In our experiments, we synthesized 18 views using the VSRS reference software for depth estimation and view synthesis tool [43] for the BreakDancers video. This creates 18 additional query videos. For the copying subset of the views case, we assume that original 3D videos can have multiple views, and only a subset of them are copied. This can be done to save storage or bandwidth of the hosting site or the viewers. In our experiments, we have two videos that have eight views each, which are Ballet and BreakDancers. For each one of them, we use view0 as the reference view, and we create 7 different queries, one for each of the remaining 7 views. Thus, we add 2*7 = 14 entries to our query videos database.

Including all transformations, **the total number of query videos in our experiments is 516.**

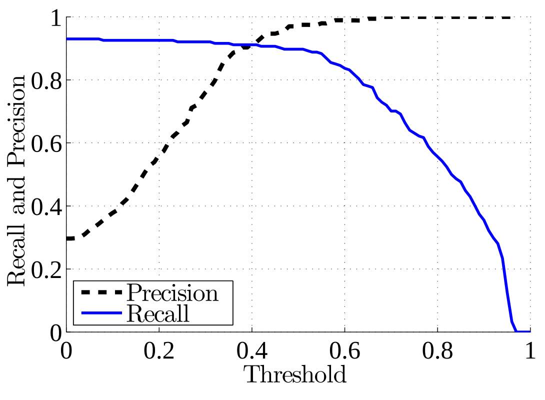
## 5.2 Experimental Setup

We conduct the following sets of experiments.

- No Transformations. Query videos are parts of some reference videos and they are not subjected to any transformations.
- Individual Transformations. Each query video is subjected to one transformation, either on the texture or depth.
- Temporal Transformations. Each query video is subjected to change in frame rate transformation, either decrease or increase.
- Combined Transformations. Each query video is subjected to multiple transformations, both on the texture and depth.
- View Synthesis and Subset of Views Transformation. A query video can contain synthesized views or a subset of the original views.
- Importance of using Depth and Visual Signatures together. Study the possibility of using depth signature only or visual signature only in the copy detection process.
- Sensitiveness of Depth signal to alternation. Showing how the depth signature and quality of the synthesized views change as the depth signal undergoes some transformations.
- Depth Signature Grid Size. Comparing the results when different grid sizes are used for the depth signature.



(a)



(b)

Figure 5.1: Precision and recall of the proposed 3D video copy detection system (a) when videos do not have any transformations and, (b) when videos are subjected to nine different transformations..

- Subsampling. Using only a subset of frames in the copy detection to speed up the process.

We consider two important performance metrics: precision and recall, which are defined in the following two equations.

$$precision = \frac{number\ of\ correctly\ identified\ copies}{total\ number\ of\ reported\ copies}. \tag{5.1}$$

$$recall = \frac{number\ of\ correctly\ identified\ copies}{actual\ number\ of\ copies}. \tag{5.2}$$

## 5.3 No Transformations

In this experiment, we evaluate the performance of the proposed 3D video copy detection system using query videos that do not have any transformations. This scenario happens, for example, when a clip is directly taken from a digital video stored on a DVD or hard disk. We compare all 48 query videos against the reference video database. We vary the threshold, which determines whether a video is a copy or not based on its score, between 0.0 and 1.0 and compute the precision and recall for each case. We plot the results in Figure 5.1(a). The figure shows that the proposed system can achieve 100% precision and

100% recall when the threshold value is between 0.7 and 0.8. For a wide range of threshold values between (between 0.5 and 0.9), the system yields 100% recall and precision.

## 5.4 Individual Transformations

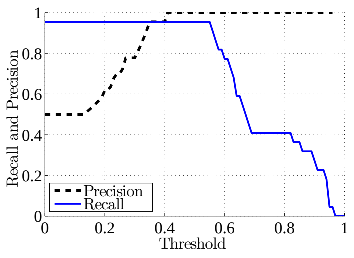
In this experiment, we apply individual transformations on query videos. This shows the robustness of the proposed copy detection system against video modifications that occur in practice when videos are copied. We apply all transformations described in Section 5.2. This means that we repeat the experiment for each transformation, and in each repetition, we vary the threshold between 0.0 and 1.0 and compute the precision and recall for each case. The results of these experiments for each transformation are shown in Figure 5.2. In all cases, the achieved precision and recall values are more than 90%, and these are obtained for a wide range of threshold values, which shows that our system does not require fine tuning of the threshold parameter. We notice that some transformations, e.g., texture blur, have more impact on the precision and recall than other transformations such as flip.

We plot the average results across all transformations in Figure 5.1(b), where we compute the average precision and recall values across all experiments for the corresponding values of the threshold. The figure shows that, on average, our system can result in 92% precision and recall at the same time (the intersection point). By using the threshold parameter, administrators can control the performance of 3D video copy detection systems based on the requirements of the system. For example, in some systems, 100% precision is desired even if some copies are not detected. For such systems, higher threshold values can be used.

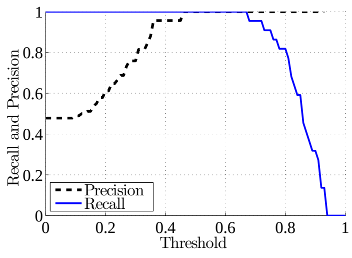
In summary, the results in this section show that the proposed system yields high precision and recall values in presence of various video transformations.

## 5.5 Multiple Transformations

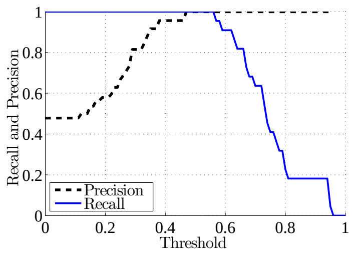
We assess the performance of our system in quite challenging scenarios, where each query is subjected to multiple transformations at the same time. We experiment with two cases. First, when each video is subjected to 3 different transformations, one applied on texture, one on depth, and one on both. The 3 transformations are chosen randomly for each query video. In the second case, we apply 5 transformations, 2 on texture, 2 on depth and one on both. We plot the average precision and recall values for these cases in Figures 5.3(a) and



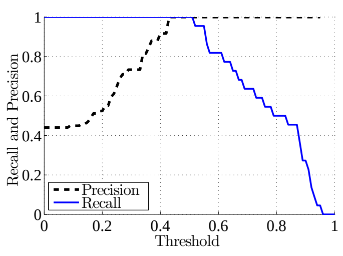
(a) Texture Blur



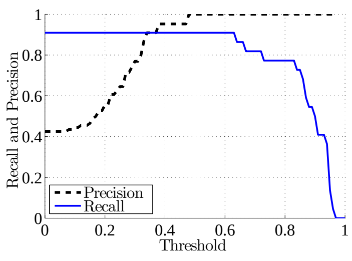
(b) Texure Gamma Correction



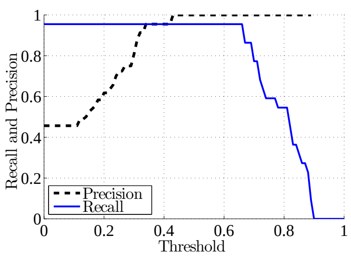
(c) Texture Noise



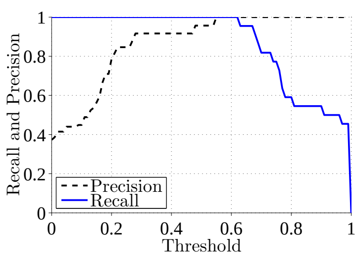
(d) Crop



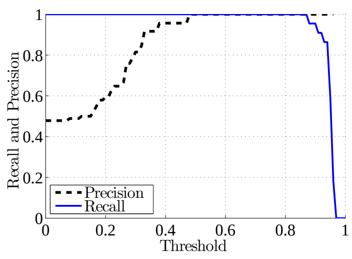
(e) Text Insertion



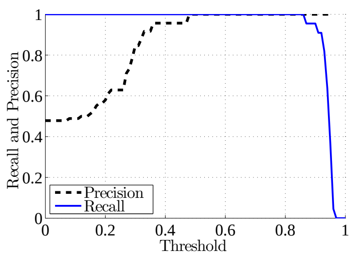
(f) Logo Insertion



(g) Flip

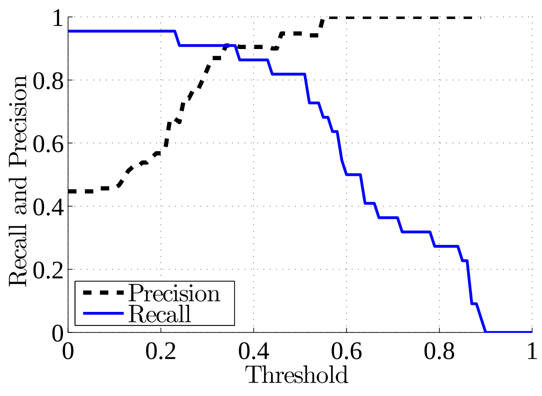


(h) Depth Blur

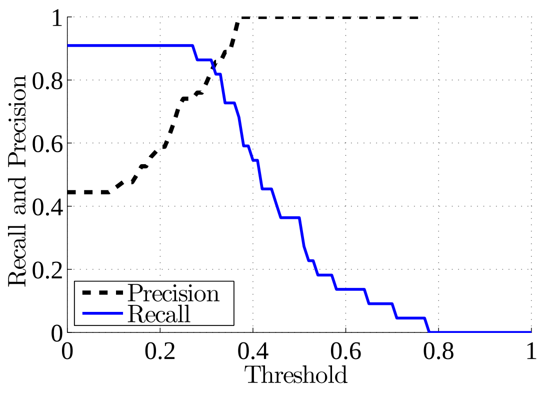


(i) Depth Noise

Figure 5.2: Precision and recall of the proposed 3D video copy detection system when videos are subjected to nine transformations.



(a) 3 Transformations



(b) 5 Transformations

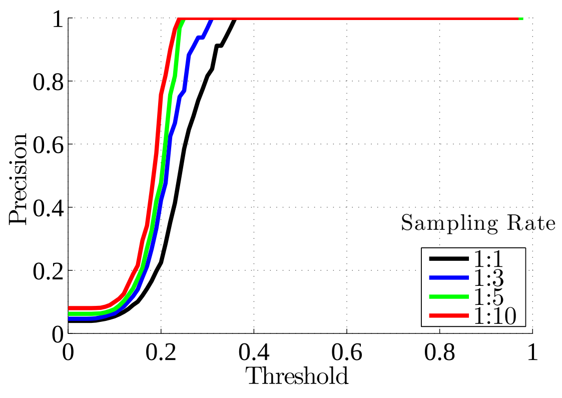
Figure 5.3: Precision and recall of the copy detection system when videos are subjected to multiple transformations.

5.3(b). The results in Figure 5.3(a) show that the proposed system achieves high precision and recall values around 90%, even when query videos are subjected to three different transformations. For extreme situations where query videos are subjected to 5 different transformations, the precision and recall values are still more than 80%. These experiments demonstrate the robustness of the proposed system to multiple video transformations.

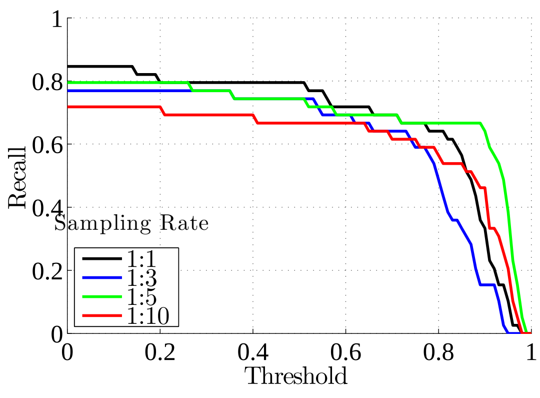
## 5.6 Subsampling

As mentioned in Section 3.2.2, subsampling in the copy detection context means that instead of extracting the signature for every frame in the video, we can extract the signature for only a subset of the frames in order to reduce the computational complexity. However, this may affect the performance of the system. So there is a trade off between the processing time and accuracy of the system.

Using the Spider system, and the reference and query videos explained in Section 5.1, different sampling rates are examined to measure the performance. For each sampling rate, all 48 queries, are processed and the total processing time, precision, and recall are used to evaluate the performance. Here a sampling ratio of $1 : X$ means we consider every $Xth$ frame in the detection process. The total processing time and the precision and recall for a fixed threshold (0.4) which the system has a good performance around it are measured. The results are shown in Figure 5.4. As shown in these figures, the recall of the system decreases



(a)



(b)

Figure 5.4: Precision and recall of the copy detection system using different sampling rate.

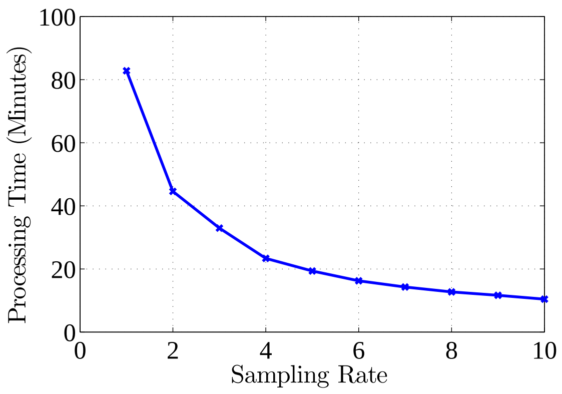


Figure 5.5: Processing time of the copy detection system, using different sampling rate.

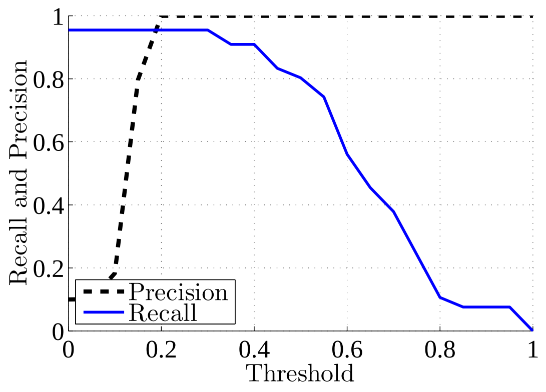


Figure 5.6: Precision and recall of the copy detection system when videos are subjected to change of frame rate transformation.

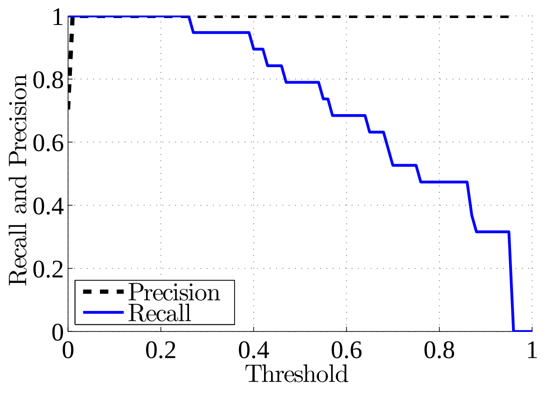
by a small amount as the sampling rate decreases, which cause the precision to increase, since some of the documents which are not declared as being a copy are actually false alarm. However, the change in precision and recall is not noticeable, while the processing time reduces significantly. This means subsampling can be used to increase the scalability of the system without reducing its performance significantly.

We note that the total query processing time of our system is spent in three main steps: depth signature extraction, visual signature extraction, and matching. From our experiment, the second step took 4.5 min, which is approximately 68% of the total processing time. The other stages took 1.6, and 0.5 minutes for first, and third steps, respectively.

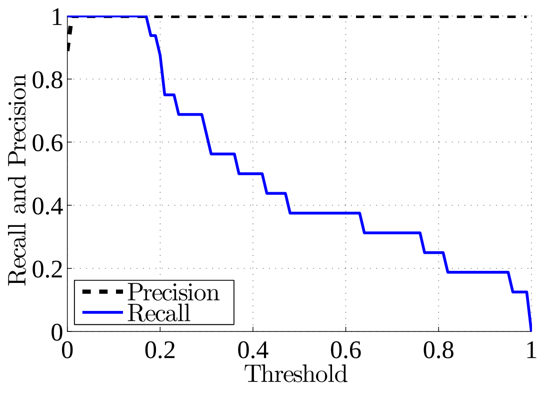
## 5.7 Temporal Transformations

In this experiment, we apply the change of frame rate transformation on query videos. To detect query videos subjected to change of frame rate transformation, we added three versions of each reference video to the index: original version, half frame rate version (every other frame were simply dropped), and twice frame rate version (every frame were duplicated). Half and twice frame rates are extreme cases, and signature of any query video subjected to frame rate change would be close enough to one of these versions.

We change the frame rate as described in Section 5.2, and plot the average precision and recall values in Figure 5.6. The system achieves high precision and recall, more than 90%, against this challenging temporal transformation.



(a) View Synthesis



(b) Subset of Views

Figure 5.7: Precision and recall of the copy detection system when there are view synthesis and subset of views transformations.

## 5.8 View Synthesis and Subset of Views

In these experiments, we apply two new transformations that are expected to appear in 3D video copy detection systems. The first one is called view synthesis, in which a virtual view is created from other views using software tools such as [43]. This is a challenging case to detect, as the synthesized views can have different viewing angles and showing objects with different depths. We compare the 18 query videos that contain synthesized views against the reference database. We vary the threshold values between 0.0 and 1.0 and compute the average precision and recall values across all queries. The results, shown in Figure 5.7(a), indicate that our system is quite robust to the view synthesis transformation as it can produce close to 100% precision and recall values. The second transformation we consider is the subset of the views transformation, in which a copied video can have a smaller number of views than the original video. We compare the 14 query views that contain subset of the views against the reference database. We plot the obtained average precision and recall values in Figures 5.7(b). The results demonstrate the robustness of our system against the subset of views transformation.

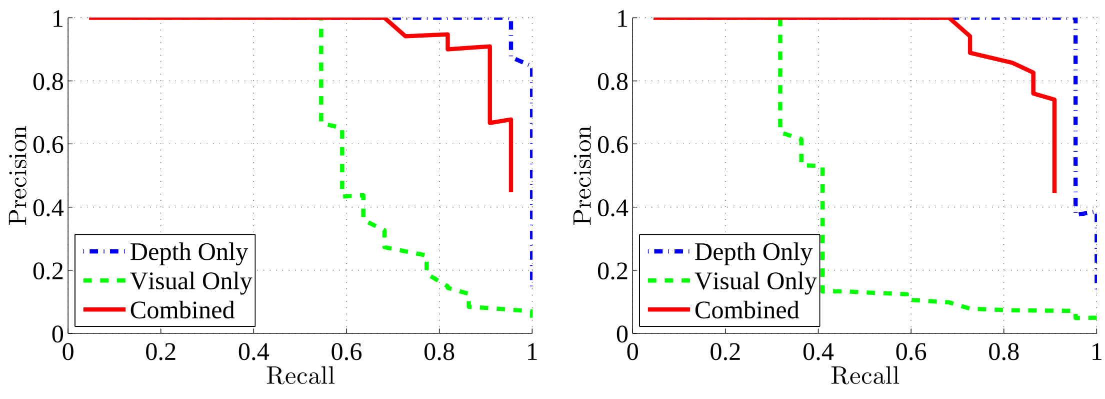
## 5.9 Importance of using Depth and Visual Signatures together

In this section, we study whether we can use either the depth signature only or the visual signature only in the 3D video copy detection system. We repeat the experiments in Section 5.5 where videos are subjected to multiple transformations at the same time. However, we use the depth signatures and visual signatures separately in the detection process. We plot the results in Figure 5.8, using precision-recall (PR) curves which expose differences between algorithms. To get the PR curve, we changed the threshold from 0 to 1, and computed the precision and recall for each threshold. Then, we plotted the recall and precision values on x and y axes. In PR space, the goal is to be in the upper-right corner. As mentioned in Section 5.1, the videos subjected to 3 and 5 transformations are in video plus depth format. The fact that depth data cannot be manipulated significantly, and that the depth signature is robust to depth manipulations, makes the depth only PR curve to be much higher than visual only curve. Clearly in this case, using depth signature, aside from fast computation, has a great benefit for the whole system. When the 3D video is not in video plus depth format (stereo or multiview), the depth is extracted from the transformed version of the views. In this case, the depth signature also degrades respectively. Figure 5.9 shows the performance of each signature, when a single transformation is applied to stereo videos, and depth is generated from the transformed version. In this case, even though combined method has lower performance in terms of Area Under the Curve (AUC), combined version has better precision for low recall range, which is desirable for retrieval systems.

## 5.10 Sensitivity of Depth Signal to Alternation

The next experiment is conducted to confirm the sensitiveness of depth videos, in the sense that altering them make the quality of the synthesized views to drop below the desired level. Two sequences, namely Ballet and Breakdancers, are used in this experiment. In these sequences, the views $v_3$ and $v_5$ are considered the original views, and these two views are used to synthesize view $\widetilde{v}_4$. For synthesizing the views, VSRS[2] software version 3.0 is used. Then, the quality of the synthesized view $\widetilde{v}_4$ is compared to the actual view $v_4$ using

[2]ISO/IEC JTC1/SC29/WG11, Reference softwares for depth estimation and view synthesis, Doc. M15377, April 2008.



(a) 3 transformations (b) 5 transformations

Figure 5.8: Precision-recall plots of the copy detection system when using depth and visual signatures separately for video plus depth 3D format videos subjected to 3 and 5 transformations.

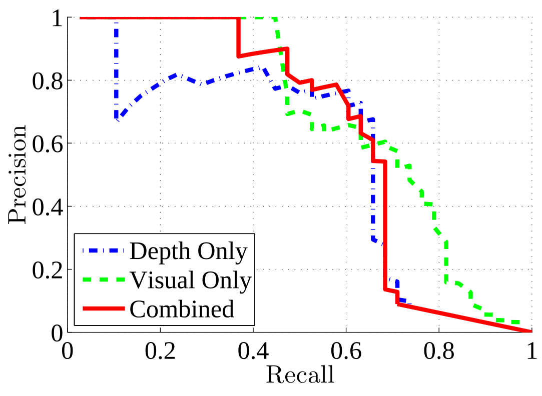
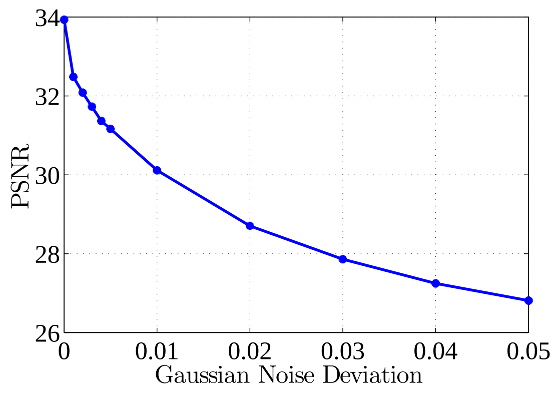
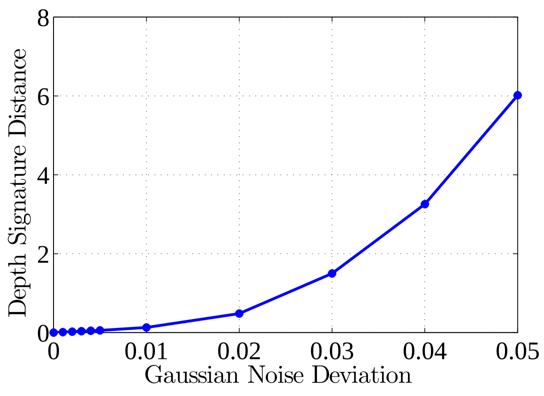


Figure 5.9: Precision-recall plots of the copy detection system when using depth and visual signatures separately for stereo 3D format videos subjected to single transformation.



(a)



(b)

Figure 5.10: Results of applying Gaussian noise to the depth images

PSNR measure. To compute the PSNR, first, the intensity values are computed using the following equation:

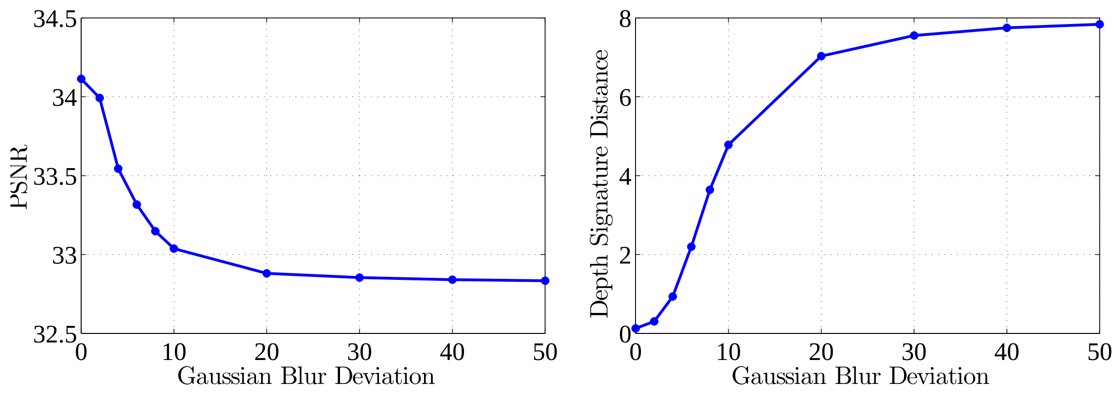
$$Y(i,j) = 0.299R(i,j) + 0.587G(i,j) + 0.114B(i,j). \tag{5.3}$$

Then, the PSNR is computed using this equation:

$$10log_{10}\frac{255^2}{(\frac{1}{W\times H})\sum_{i,j=1}^{i=W,j=H}(Y(i,j)-\widetilde{Y}(i,j))^2}. \tag{5.4}$$

In Figure 5.10, Gaussian noise with increasing deviation is applied to the depth images, while the texture is remained untouched. Then, the virtual view is synthesized using the untouched texture and degraded depth. As it can be seen in Figure 5.10(a) even with small amount of noise with deviation of 0.01, the quality of the synthesized view drop about 4db. An example of the synthesized view using the noisy depth of deviation 0.01 is shown in the left side of Figure 5.12. However, as it is shown in Figure 5.10(b), the depth signature distance of the noisy depth videos from the original one has not change much. In other words, the depth signature distance of the noisy depth with deviation of 0.05 from the original view is around 6, while the depth signature distance of a random video would be above 1200.

In Figure 5.11, the alternation is blurring. Blurring the depth images is a method that is actually proposed in the literature to improve the quality of the synthesized views, sine



(a) (b)

Figure 5.11: Results of applying gaussian blurring to the depth images.

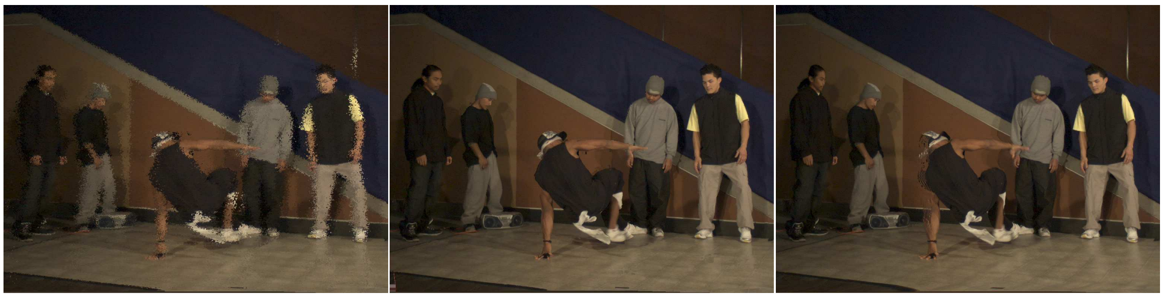


Figure 5.12: The left image is synthesized using the noised depth image. The center image is the synthesized view from the original depth image, and the right image is the synthesized view using the blured depth image.

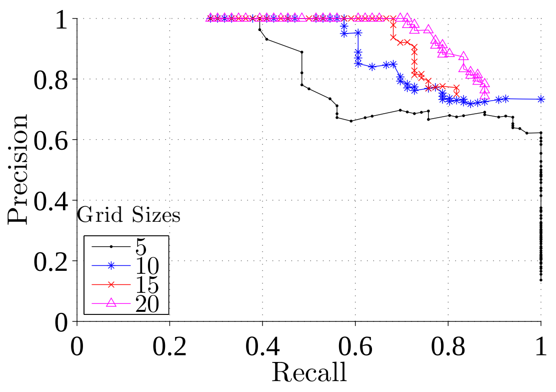


Figure 5.13: Precision-recall diagrams for four grid sizes.

blurring makes the depth images smoother, so there would be less disocclusion effects near depth discontinuities or edges. However, blurring causes quality reduction in the areas that have a constant depth, and make their texture to become inaccurate. That is why new algorithms have been proposed to do blurring only along the depth discontinuities [5]. As it can be seen in Figure 5.10(a) with a Gaussian blurring with deviation of 0.1, the quality of the synthesized view drops about 1db. However, as it is shown in Figure 5.10(b), the depth signature distance of the blurred depth videos from the original one has not change much. In other words, the depth signature distance of the blurred depth with deviation of 50 from the original view is around 8, while, as mentioned before, the depth signature distance of a random video would be above 1200. An example of the synthesized view with depth blurred by deviation of 0.05 is shown in the right side of the Figure 5.12.

## 5.11 Depth Signature Grid Size

In all experiments discussed earlier, the depth signature was obtained by dividing the depth signal to a 20x20 grid. In this experiment, it is shown why this grid is chosen. To achieve this, the gird size for depth signature must be altered to different values, and the depth signature of the reference and query videos must be computed for each size. To compare the performance, the results for grids 5x5, 10x10, 15x15, and 20x20 are shown in Figure 5.13. To plot the precision-recall diagrams shown in these figures, only depth signatures are used, since the grid size is independent of visual part. Also, to compute the precision and recall, the results for 3 cases are considered together. These cases are when there is no transformation to the depth, when the depth images are blurred, and when some noise are added to the depth images. It is clear that for a specific recall, the precision is higher when the grid is 20x20. In general we can say that the number of blocks in the grid trades off the computational complexity and the copy detection accuracy.

Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

Three dimensional (3D) videos are getting quite popular, and creating 3D videos is expensive. Thus, protecting 3D videos against illegal copying is an important problem.

We presented the detailed design of a 3D video copy detection system. The system has two main components: Processing Reference Videos and Processing Query Videos. In the first component, the system creates compact signatures of the depth and texture of the reference videos and store them in a database. The second component creates similar signatures for each query video and compares them against signatures in the database. If a match is found, the location of the copied part in the reference video is also identified.

We implemented the proposed system and evaluated its performance in terms of precision and recall using many 3D videos. Some of these videos have two views, where the others have eight different views. We created a large set of query videos, which has a total of 516 3D videos. We carefully customized the query videos to represent most practical scenarios for copying 3D videos. Specifically, our query videos represent the following scenarios: (i) query videos are segments of some reference videos, (ii) each query video is subjected to nine different transformations, either on the texture or depth, (iii) multiple combined transformations are applied on the texture and depth of each video, (iv) new views are synthesized from existing ones, and (v) query videos have only a subset of views of reference videos.

Our experimental results show that the proposed system achieves high precision and recall values in all scenarios. Specifically, the proposed system results in 100% precision and recall when copied videos are unmodified parts of original videos, and it produces more than 90% precision and recall when copied videos are subjected to different individual transformations. Even in the extreme cases where each video is subjected to five different transformations at the same time, our system yields precision and recall values more than 75%. Furthermore, the above results are obatained for a wide range of the threshold parameter used in the system, which means that our system does not need fine tuning of that parameter.

## 6.2 Future Works

Audio is as an inseparable component of videos. Similar to texture and depth signals, audio signal can be subjected to transformations such as MP3 compression, and mixing with speech. Sometimes the audio signal is completely replaced, for example when movies are translated to other languages, so we cannot rely only on audio signals. Various audio signatures have been proposed in the literature [4] [12]. In the future, audio signatures can be used along side with visual and depth signatures to refine the results and achieve higher precision and recall.

As mentioned before, the signatures can be created for every frame in the video, or for only a subset of the frames in order to reduce the computational complexity. In Section 5.6, where subsampling of frames were examined, this subset of the frames was chosen deterministically, e.g., each 5th frame was chosen. In the future, the subset of the frames can be the keyframes of the video, where a keyframe is a representative frame for a sequence of video frames containing similar visual information, which is referred to as a video shot. Shot boundary detection algorithms such as [22] can be employed to determine when a shot starts and ends. Keyframe selection algorithms such as [11] can be used to select key frames.

The current implementation runs on a single machine. However, the system has a great potential of being implemented in a distributed environment to speedup the process. For example, signature extraction and nearest neighbor search of each frame is independent of other frames, so it can be performed on separate machines. Especially, we suggest the use of cloud computing, since the workload of the system is not steady, and goes up as the number of videos and their frames that needs to be checked increases. So the system can benefit from the fine grain (one server at a time with Amazon EC2[1]) scalability which can be ready in terms of minutes rather than weeks, so we can fit the workload to resources efficiently and in a pay-as-you-use manner.

There are several services with similar technology to video copy detection. Examples of these services are:

- Broadcast monitoring: Using fingerprint technology to determine when, where, and how a specific content has been used to measure the effectiveness of the content.
- Content aware advertisement: Using automated content recognition based on perceptual fingerprinting to find potential customers. It is very important for this service to act on real time, since it will not be useful if it cannot react in real time to what a person is watching.
- Metadata services: Music and movies metadata provide information on song titles, artists, albums, record labels, and other related content. Using the content fingerprints to provide the metadata to the user improves the user experience.

The proposed system can be expanded to cover these services as well.

[1]`http://aws.amazon.com/ec2/`

# Bibliography

[1] M. Aly, M. Munich, and P. Perona. Distributed Kd-Trees for Retrieval from Very Large Image Collections. In *Proc. of British Machine Vision Conference (BMVC'11)*, pages 40.1–40.11, Dundee, UK, August 2011.

[2] M. Aly, M. Munich, and P. Perona. Indexing in large scale image collections: Scaling properties and benchmark. In *Proc. of IEEE Workshop on Applications of Computer Vision (WACV'11)*, pages 418–425, HI, USA, January 2011.

[3] Bino Free 3D Player. `http://bino3d.org/`.

[4] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41(3):271–284, November 2005.

[5] W. Y. Chen, Y. L. Chang, S. F. Lin, L. F. Ding, and L. G. Chen. Efficient depth image based rendering with edge dependent depth filter and interpolation. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'05)*, pages 1314–1317, Amsterdam, Netherlands., July 2005.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of Twentieth Annual Symposium on Computational geometry (SCG'04)*, pages 253–262, Brooklyn, New York, USA, June 2004.

[7] N.A. Dodgson. Autostereoscopic 3D displays. *Computer*, 38(8):31–36, August 2005.

[8] FFmpeg. `http://www.ffmpeg.org/`.

[9] FLANN - Fast Library for Approximate Nearest Neighbors, June 2011. `http://www.cs.ubc.ca/~mariusm/index.php/FLANN`.

[10] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.*, 3(3):209–226, September 1977.

[11] Y. Gong and X. Liu. Video summarization using singular value decomposition. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'00)*, pages 174–180, Hilton Head, SC, USA, June 2000.

[12] V. Gupta, G. Boulianne, and P. Cardinal. Crim's content-based audio copy detection system for trecvid 2009. In *Proc. of Internation Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6, June 2010.

[13] A. Hampapur, K. Hyun, and R. M. Bolle. Comparison of sequence matching techniques for video copy detection. In *Proc. of SPIE Conference on Storage and Retrieval for Media Databases (SPIE'02)*, pages 194–201, San Jose, CA, USA, January 2002.

[14] R. C. Harvey and M. Hefeeda. Spatio-temporal video copy detection. In *Proc. of ACM Multimedia Systems (MMSys'12)*, pages 35–46, Chapel Hill, NC, USA, February 2012.

[15] Jopensurf. `http://code.google.com/p/jopensurf`.

[16] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe. Watermarking techniques for intellectual property protection. In *Proc. of the 35th Annual Design Automation Conference (DAC'98)*, pages 776–781, San Francisco, CA, US, June 1998.

[17] P. Kauff, N. Atzpadin, C. Fehn, M. Muller, O. Schreer, A. Smolic, and R. Tanger. Depth map creation and image-based rendering for advanced 3DTV services providing interoperability and scalability. *Signal Processing: Image Communication*, 22(2):217–234, February 2007.

[18] N. Khodabakhshi and M. Hefeeda. Copy detection of 3D videos. In *Proc. of ACM Multimedia Systems (MMSys'12)*, pages 131–142, Chapel Hill, NC, USA, February 2012.

[19] N. Khodabakhshi and M. Hefeeda. Spider: A system for finding 3D video copies. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2012.

[20] A. Koz, C. Cigla, and A.A. Alatan. Watermarking of free-view video. *IEEE Transactions on Image Processing*, 19(7):1785–1797, July 2010.

[21] Z. Li and J. Chen. Efficient compressed domain video copy detection. In *Proc. of International Conference on Management and Service Science (MASS'10)*, pages 1–4, Wuhan, China, August 2010.

[22] Z. Liu, D. Gibbon, E. Zavesky, B. Shahraray, and P. Haffner. A fast, comprehensive shot boundary determination system. In *Proc. of IEEE International Conference on Multimedia and Expo (ICME'07)*, pages 1487–1490, Beijing, China, July 2007.

[23] Z. Liu, T. Liu, D. C. Gibbon, and B. Shahraray. Effective and scalable video copy detection. In *Proc. of the international conference on Multimedia information retrieval (MIR'10)*, pages 119–128, Philadelphia, Pennsylvania, USA, 2010.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, November 2004.

[25] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, April 2008.

[26] P. Merkle, K. Muller, and T. Wiegand. 3D video: Acquisition, coding, and display. In *Proc. of Digest of Technical Papers International Conference on Consumer Electronics (ICCE'10)*, pages 127–128, January 2010.

[27] P. Merkle, A. Smolic, K. Muller, and T. Wiegand. Efficient prediction structures for multiview video coding. *Circuits and Systems for Video Technology, IEEE Transactions on*, 17(11):1461–1473, November 2007.

[28] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligenc*, 27(10):1615 –1630, October 2005.

[29] Mobile 3DTV. `http://sp.cs.tut.fi/mobile3dtv/video-plus-depth/`.

[30] Y.i Mori, N. Fukushima, T. Yendo, T. Fujii, and M. Tanimoto. View generation with 3D warping using depth information for FTV. *Signal Processing: Image Communication*, 24(1-2):65–72, March 2009.

[31] Microsoft research, MSR 3D Video. `http://research.microsoft.com/en-us/um/people/sbkang/3dvideodownload/`.

[32] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proc. of International Conference on Computer Vision Theory and Applications (VISAPP'09)*, pages 331–340.

[33] V. Ramachandra, M. Zwicker, and Truong Nguyen. 3D video fingerprinting. In *Proc. of 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV'08)*, pages 81–84, Istanbul, Turkey, May 2008.

[34] G. Roth, R. Laganie andre, P. Lambert, I. Lakhmiri, and T. Janati. A simple but effective approach to video copy detection. In *Proc. of Canadian Conference on Computer and Robot Vision (CRV'10)*, pages 63–70, Ottawa, Ontario, Canada, June 2010.

[35] H. Y. Shum, Y. Li, and S. B. Kang. An introduction to image-based rendering. In David Zhang, Mohamed Kamel, and George Baciu, editors, *Integrated Image and Graphics Technologies*, volume 762 of *The Kluwer International Series in Engineering and Computer Science*. Springer Netherlands, February 2004.

[36] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'08)*, pages 1–8, Anchorage, Alaska, USA, June 2008.

[37] A. Smolic, K. Müller, N. Stefanoski, J. Ostermann, A. Gotchev, G. B. Akar, G. Triantafyllidis, and A. Koz. Coding algorithms for 3DTV-a survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11):1606–1621, November 2007.

[38] R. Szeliski. Stereo correspondence. In David Gries and Fred B. Schneider, editors, *Computer Vision*, Texts in Computer Science, pages 467–503. Springer London.

[39] R. Szeliski. Feature detection and matching. In David Gries and Fred B. Schneider, editors, *Computer Vision*, Texts in Computer Science, pages 181–234. Springer London, 2011.

[40] K. Tasdemir and A.E. Cetin. Motion vector based features for content based video copy detection. In *Proc. of International Conference on Pattern Recognition (ICPR'10)*, pages 3134–3137, Istanbul, Turkey, August 2010.

[41] Traixes DepthGate. `http://doc.triaxes.tv/depthgate`.

[42] T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, July 2008.

[43] ISO/IEC JTC1/SC29/WG11, Reference softwares for depth estimation and view synthesis. Doc. M15377, April 2008.

[44] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programming. In *Proc. of Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 798–805, North Carolina, USA, June 2006.

[45] J. You, L. Xing, A. Perkis, and X. Wang. Perceptual quality assessment for stereoscopic images based on 2D image quality metrics and disparity analysis. In *Proc. of International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM'10)*, pages 1–6, Scottsdale, AZ, USA, January 2010.

[46] Z. Zhang and J. Zou. copy detection based on edge analysis. In *Proc. of IEEE International Conference on Information and Automation (ICIA'10)*, pages 2497–2501, Harbin, Heilongjiang, China, June 2010.

[47] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 23(3):600–608, August 2004.