Copy Detection of 3D Videos

Naghmeh Khodabakhshi[*]
School of Computing Science
Simon Fraser University
Surrey, BC, Canada
nkhodaba@sfu.ca

Mohamed Hefeeda
Qatar Computing Research Institute
Qatar Foundation
Doha, Qatar
mhefeeda@qf.org.qa

ABSTRACT

We present a novel system to detect copies of 3D videos. The system creates signatures from the depth signals of 3D videos. It also extracts visual features from video frames and creates compact spatial signatures for videos. The system then uses the depth and spatial signatures to compare a given query video versus a reference video database. The system returns a score indicating whether the query video matches any video in the reference video database, and in case of matching, which portion of the reference video matches the query video. The system is computationally efficient and can be implemented in distributed manner. The system can be used, for example, by video content owners, video hosting sites, and third-party companies to find illegally copied 3D videos. To the best of our knowledge, this is the first complete 3D video copy detection system in the literature. We implemented the proposed system and conducted a rigorous evaluation study using 3D videos with diverse properties. 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.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering, Search process*

General Terms

Algorithms, Experimentation

[*]This work was done during an internship in Qatar Computing Research Institute (QCRI).

Keywords

Video copy detection, 3D video, Video fingerprinting, Depth features, Visual features

1. INTRODUCTION

Many experts are anticipating that the future of video is three dimensional (3D). This is backed up by the phenomenal success of 3D videos such as Avatar. Other notable examples include broadcasting of the vastly popular 2010 FIFA World Cup in 3D. 3D videos provide more realistic perception of displayed objects on the screen, and thus they increase viewers' enjoyment. YouTube is currently hosting 3D videos; there are more than 14000 videos labeled with 3D tag at the time of writing this paper. With rapid advances in 3D cameras and displays, numerous 3D videos are expected to be created and consumed in the near future. Since 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. One of these protection schemes is to detect illegally created copies of 3D videos, which is the focus of this paper.

Detecting copies of traditional 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 very computationally intensive. Detection of 2D 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. 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 during copying.

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 many more 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 [21]. 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.

In this paper, we present a novel system to detect 3D video copies. To the best of our knowledge, this is the first, content-based, complete 3D video copy detection system in the literature. The proposed system creates signatures from the depth and texture signals of 3D videos. These signatures are compact and robust to many 3D video transformations, including the ones described above. We implemented the proposed system and conducted a rigorous evaluation study using many 3D videos that have diverse characteristics and are encoded in different formats. 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.

The rest of this paper is organized as follows. In Section 2, we summarize the related works in the literature. In Section 3, we present the details of the proposed 3D video copy detection system. We present our extensive experimental evaluation in Section 4, and we conclude the paper in Section 5.

2. RELATED WORK

There are different methods for video copy detection. One method called watermarking [9] 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.

3D watermarking approaches in the literature can be classified into three groups [23]: (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 [23]. While the first two groups have been studied quite widely, the third group emerged after image based rendering techniques developed [11].

Alper et al. [11] 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. [19] where they propose a method to protect multiview 3D videos using a fingerprint based on scale invariant feature transform (SIFT) [15], 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. [8] 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. [25] use the motion vectors for the signature of a frame. Some other methods [12] [28] use fingerprints which are obtained directly from compressed videos. Another group of methods use color histograms as videos' fingerprint. For instance, [8] 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. [14] use local features that are extracted by SIFT as the frame signature. Roth et al. [20] take every frame of the source video and divide it into 16 regions. They then use Speeded-Up Robust Features (SURF) [16] to find local points of interest in the frame.

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 4.7.

3. PROPOSED 3D VIDEO COPY DETECTION SYSTEM

In this section, we start by presenting an overview of the proposed 3D video copy detection system and how it can be used. Then, we present the details of its different components. Then, we analyze space and time complexity of the system.

3.1 Overview

We propose a novel system to detect 3D video copies. Figure 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.

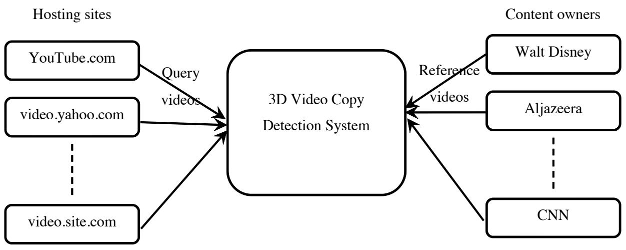


Figure 1: High-level illustration of the video copy detection system.

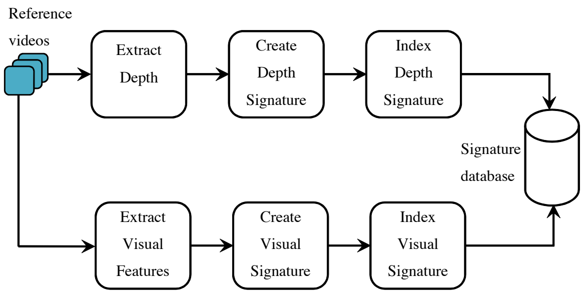


Figure 2: Steps for processing reference videos

- 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.

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, Processing Reference Videos, is summarized in Figure 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

The depth in a 3D video is emulated by presenting two slightly different views to the human eyes. The human brain fuses these two views to perceive the third dimension. Depending on the display technology, goggles may be needed to control the view seen by each eye and at what time. Different methods exist for preparing and coding 3D videos:

- Stereo Video. The video has two views. A view can be thought of as a separate 2D 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 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 [10] and [17]. 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.

For 3D videos encoded in video plus depth format (or its variants), the depth information for each video 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 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. [24] 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 are available to generate depth information in real-time [27].

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, 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 20×20 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 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 [13] can be employed to identify when a shot starts and ends. Keyframe selection algorithms such as [7] 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 neighbour search efficiently. Based on the requirements of the system, like the performance or the need to be easily distributable, a method can be chosen. We propose to use LSH for nearest neighbor search for large-scale video copy detection system, because it can be easily parallelized. For small-scale systems, randomized kd-tree [22] is sufficient.

We present a brief background on LSH, and then show how it can be used in our system. The basic idea of LSH [6] is to hash high dimensional vectors to integer hash values in a way that when the vectors are close to each other in the original space, their hash values are likely to be close to each other as well.

LSH employs a family of hash functions, which is denoted by $H = \{S \to U\}$, where S is the set of input points and U is the set of hash values. A hash function $h \in H$ is called (r_1, r_2, p_1, p_2) sensitive for a distance measure D, which can be Euclidean distance, if for any $v, q \in S$, we have:

$$v \in B(q, r_1),\ then Pr_H[h(v) = h(q)] \geq p_1, \text{ and}$$

$$v \notin B(q, r_2), \text{ then } Pr_H[h(v) = h(q)] \leq p_2$$

where $B(q, r)$ stands for a sphere of radius r centered at q. For the hash function to be useful, we should have $r_1 < r_2$, and $p_1 > p_2$. To amplify the gap between p_1 and p_2, several functions are used and the output hashes are concatenated. In other words, a new family of functions are defined as $G = \{S \to U^k\}$, $g(v) = (h_1(v), h_1(v), ..., h_k(v))$ where $h_i \in H$. Then, L functions from the G family are chosen independently and uniformly at random for hashing.

In order to index depth signatures, we perform the following steps;

- Choose L functions from the G family.
- Consider each depth signature, which corresponds to a video frame, as one data point v.
- Apply the chosen functions on v. The result is L entries in various buckets of the index. Each entry in a bucket has the following fields: $< VideoID, FrameID, ViewID, DepthSignature >$
- To find nearest neighbours of a query depth signature, the entries of the L buckets of its L hashed values are searched, and points close to the query depth signature are returned.

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 [16] and SIFT [15]. In our implementation, we use SIFT features, which are proved to be robust against transformations compared to other descriptors [18].

3.2.5 Create Visual Signature

The number of visual features extracted from each video frame can be controlled by configuring the feature extraction algorithm. Reducing the number of extracted features results in reduction in the computational complexity of the copy detection system, but might introduce some errors in the detection process. For SIFT features, we set the peak threshold and edge threshold parameters of the SIFT feature extraction algorithm to control the number of SIFT features. The visual signature for a video frame takes the form $< v_1, v_2, ..., v_V >$, where V is the total number of visual features in a frame, and v_i is the value of visual feature i. We note that each visual feature v_i has multiple elements. For example, a SIFT feature usually has 128 elements.

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, multiple methods can be used to achieve this nearest neighbor search efficiently, including but not limited to locality sensitive hashing.

Similar to depth indexing, each visual signature is hashed to L' buckets, and an entry is stored for it in each bucket. This entry has the following fields: $< VideoID, FrameID, ViewID, FeatureID, VisualSignature >$. Then, to find nearest neighbors of a query visual signature, the entries of the L' buckets of its hashed values are searched.

3.3 Processing Query Videos

The second component of the proposed system is Comparing Query Videos, which is summarized in Figure 3. The depth signature is first computed from the query video. The methods used to extract depth information and create depth signatures are the same as the ones used to process reference videos. Then, the depth signature of the query video is compared against the depth signatures in the reference video database. If there is no match, the query video is not considered for any further processing. If a match occurs, the visual signature is computed from the query video and compared against visual signatures in the reference video database. Then, 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. This method is computationally efficient as it eliminates many query videos by checking their depth signatures first, which are more compact and faster to compare than visual signatures. 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.

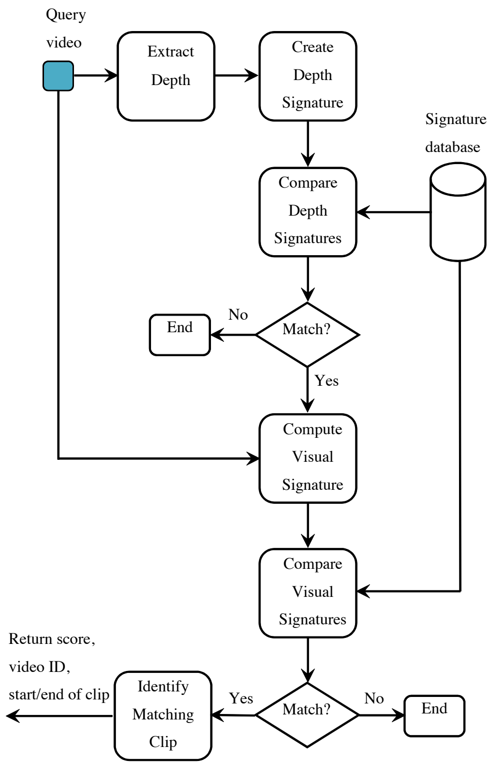


Figure 3: Comparing query video against reference video.

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 a their Euclidean distance are found using the nearest neighbor search method. Using LSH , the L buckets of the hashed values of the depth signature of the query video frame are identified. Then, the distances between the query depth signature and all depth signatures in those buckets are computed. 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. By temporal characteristics we mean that 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 of the frames. Figure 4 shows an example, where dark squares represent matched frames. Using this matrix, the longest diagonal sequence with the largest number of matching frames is considered as a potential copy.

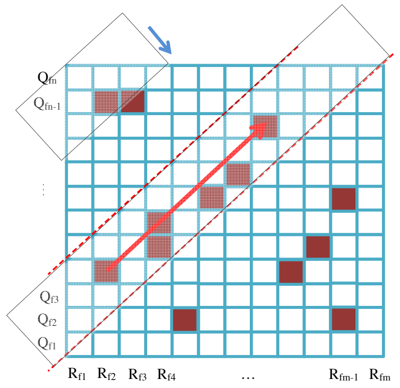


Figure 4: Matching matrix for frames from query and reference videos considering their timing.

It is worth mentioning that frame dropping and occasional frame mismatches caused by possible transformations must be taken into account. Thus, the diagonal sequence mentioned before is not a strictly linear one and gaps may exist. To find the longest diagonal sequence with the greatest score, instead of considering a line of frames, a band with a specific width is considered, as shown in Figure 4. This band starts sweeping the matrix from top left most position and moves one block each time. At each position, the temporal score of the longest diagonal 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 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 LSH, the L' buckets of the hashed values of the visual signature vector $< v_1, v_2, ..., v_V >$ are searched and the founded entries which look like this $< VideoID, FrameID, ViewID,$ $FeatureID$, $VisualSignature >$ are returned. These returned features may belong to different frames of different videos. To find the matched frames, the number of matched features in each reference frame is counted, and frames with the number of matched features exceeding a threshold are considered a match. Then, a frame level matching is performed like the one explained for the depth signature. 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 4 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 in 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 an LSH index to store signatures in order to facilitate nearest neighbor search. We note that an LSH index for n data points takes a space of $O(n^{1+\rho})$, where $\rho(c) < 1/c$ for $c \in (1, 10]$ [6]. When c is 1 the problem is exact nearest neighbor, which is not necessary for our task. To consider the worst case, for analysis, the value of c is considered close to 1. Thus, $\rho(c) = 1$ In our system, the indexed data points are the depth signatures, as well as the visual signatures, which are computed from individual frames of reference videos. Let the total number of frames in all reference videos be N_r. Since we have constant number of depth and visual signatures per frame, the space needed to store all signatures in an LSH is $O(N_r^2)$. This amount of space, considering the fact that signatures of a frame are extremely compact compared to the actual frames, makes the system practical for real world purposes.

Next, we analyze the time complexity, which is the time needed to process a given 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. In this case, all steps of the detection algorithm are executed as explained in Figure 3, which includes computing and comparing depth signatures, computing and comparing visual signatures, and determining the location of the copied clip in the reference video. Other cases take much less time, because the algorithm can terminate before executing all steps. We assume that the time taken to compute the depth signature for a frame is T_d, and to compute the visual signature is T_v. T_d and T_v do not depend on the number of frames in the query video N_q. Other computational steps, namely comparing signatures and identifying location of copied clip, do depend on N_q. For comparison of signatures, we use LSH, which takes $O(N_r logN_r)$ to compare one signature against the signature database. Thus, for comparing the depth signature, the query video that has N_q frames, we need $O(N_q N_r logN_r)$ steps.

Assuming the depth matching step returned a fixed number of potential copies, say K reference videos, we perform the visual signature comparison of those K reference videos. We compare these K reference videos against the query video, constructing the matching matrix, such as the one shown in Figure 4 between the query video and the reference video, which takes $O(LKN_q)$ steps, where L is a constant referring to the number of frames in the longest video.

Therefore, the worst case running time to process a query video of length N_q is $O(N_q N_r logN_r)$, which means given a reference database of fixed size, the running time of the system grows linearly with the number of frames considered for processing in the query video. Also, the running time grows with $O(N_r logN_r)$ on the total number of frames of reference videos in the database, which makes the system scalable to large reference databases.

Using an IntelXeon W3530 2.8 GHz processor, and the experimental setup described in section 4.2, the processing time of a 320×240 query video with 905 frames, took 56 seconds in total. From these 56 seconds, depth signature extraction, visual signature extraction, and matching took 11, 30, and 15 seconds, respectively.

4. EXPERIMENTAL EVALUATION

We have implemented the proposed 3D video copy detection algorithm. We conducted extensive analysis using real 3D videos to assess its performance. We start by describing how we prepared our video dataset in the following section. Then, we describe our implementation and experimental setup. Then, we present the results of our different experiments to show the performance of the proposed system.

4.1 Preparing 3D Video Dataset

We collected 17 3D videos with different characteristics from two sources. All videos have texture and depth signals. 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 [2]. Depth maps are computed using the method described in [29]. The other 15 videos are obtained from the MOBILE3DTV project [3]. 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 [1].

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 16 reference videos, which are listed in the first 16 rows in Table 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 1:

- Type 1: Query videos are segments or clips of reference videos. These are the first 16 rows in Table 1.
- Type 2: Query videos are segments of reference videos embedded in other videos. These are rows 17 to 22 in Table 1.
- Type 3: Query videos contain no parts of the reference videos. These are the last 6 rows in Table 1.

Then, we apply different video transformations on the 28 videos shown in Table 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, or 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.

We apply the following 9 transformations on each of the 28 videos in Table 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 deviation 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].

Table 1: List of videos used in creating query videos.

Query	Resolution	nFrames	InsertPoint
Alt-moabit	432×240	100	1-100
Ballet	1024×768	100	1-100
Book-arrival	512×384	100	1-100
BreakDancers	1024×768	100	1-100
Car	480×270	235	1-235
Caterpillar	480×270	101	1-101
Door-flowers	512×384	100	1-100
Flower1	480×270	152	1-152
Flower2	480×270	234	1-234
Flower3	480×270	112	1-112
Grasshopper	480×270	181	1-181
Hands	480×270	251	1-251
Horse	480×270	140	1-140
Leaving-Laptop	512×384	100	1-100
Rollerblade	320×240	905	1-905
Snail	480×270	189	1-189
Berlin-Ballet	Mix	132	21-81
Berlin-Break	Mix	142	31-111
Berlin-DoorFlowers	Mix	92	26-86
Berlin-Flower2	Mix	122	11-111
Berlin-Grasshopper	Mix	142	11-101
Berlin-Snail	Mix	92	11-70
TU-Berlin1	360×288	150	-
TU-Berlin2	360×288	149	-
TU-Berlin3	360×288	149	-
TU-Berlin4	360×288	149	-
TU-Berlin5	360×288	149	-
TU-Berlin6	360×288	149	-

- 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 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 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.

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 [4] 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.

Therefore, including all transformations, the total number of query videos in our experiments is 284.

4.2 Implementation and Experimental Setup

We implemented all steps of the proposed 3D video copy detection system as described in Section 3. We modified and integrated a few open-source libraries in our system. We provide some highlights of our implementation in the following.

The depth signature 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. In the second step, the depth signature is indexed to facilitate nearest neighbor search needed in the query processing phase. The index is created using the FLANN library version 1.6.11 [5]. We use the kd-tree implementation of this library in our experiments. For the visual signature, we extract SIFT features using VLFeat[26], which is implemented in C and it has Matlab interface as well. There are two parameters that need to be set in this library: peak threshold and edge threshold. We set both to 5. On average, this results in about 200 SIFT features per frame. This library returns 128 non-negative values for each SIFT feature, which we normalize to the unit vector to reduce the effects of contrast and gain. After extracting the SIFT features, we index them using the FLANN library.

We conduct the following sets of experiments, which are described in later sections.

- 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.

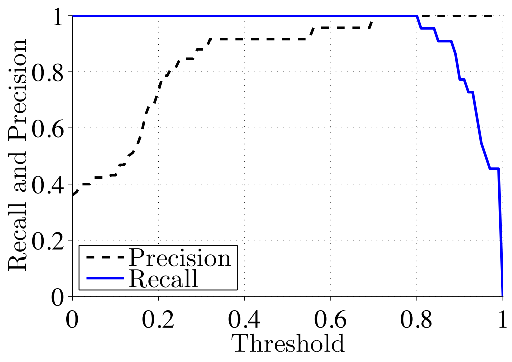


Figure 5: Precision and recall of the proposed 3D video copy detection system when videos do not have any transformations.

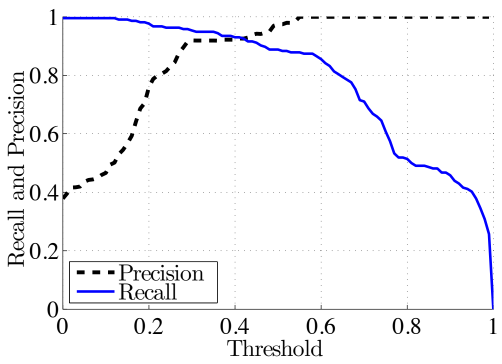


Figure 6: Average precision and recall of the proposed 3D video copy detection system when videos are subjected to nine different transformations.

- Combined Transformations. Each query video is subjected to multiple transformations, both on the texture and depth.

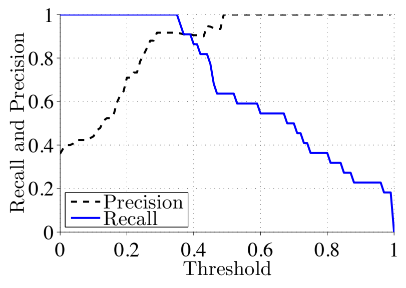
- View Synthesis and Subset of Views Transformation. A query video can contain synthesized views or a subset of the original views.

- 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.

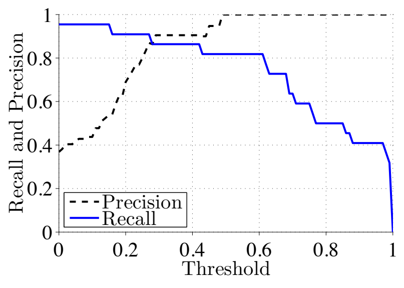
We consider two important performance metrics: precision and recall, which are defined in the following two equations.

$$precision = \frac{number\ of\ correctly\ identified\ copies}{total\ number\ of\ reported\ copies}. \quad (1)$$

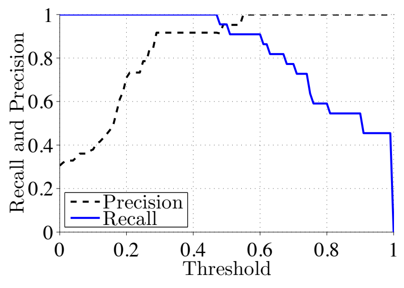
$$recall = \frac{number\ of\ correctly\ identified\ copies}{actual\ number\ of\ copies}. \quad (2)$$



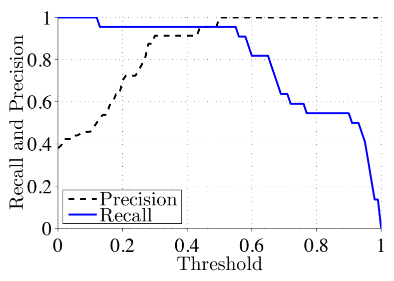
(a) Texture Blur



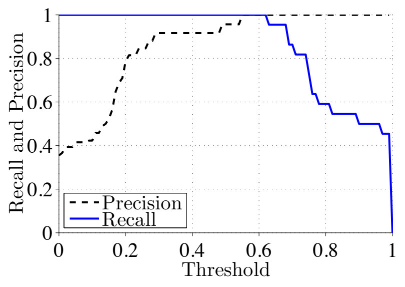
(b) Texure Gamma Correction



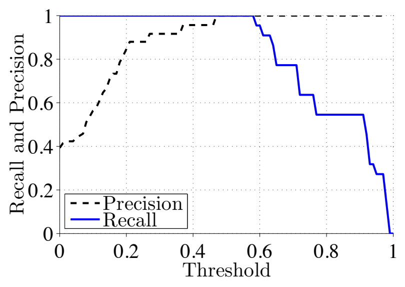
(c) Texture Noise



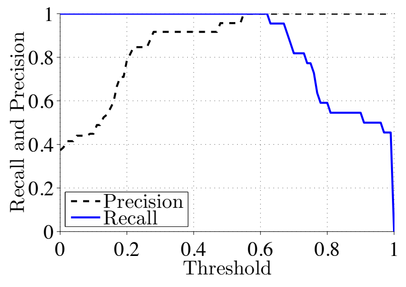
(d) Crop



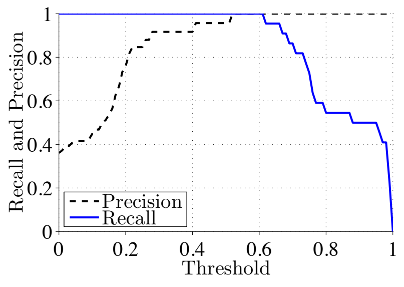
(e) Text Insertion



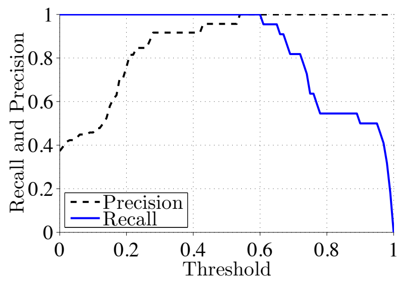
(f) Logo Insertion



(g) Flip

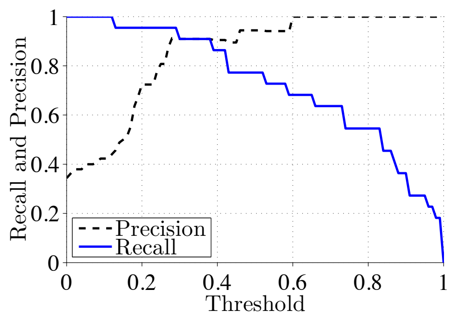


(h) Depth Blur

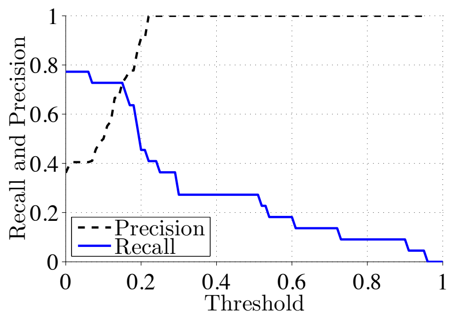


(i) Depth Noise

Figure 7: Precision and recall of the proposed 3D video copy detection system when videos are subjected to nine transformations.

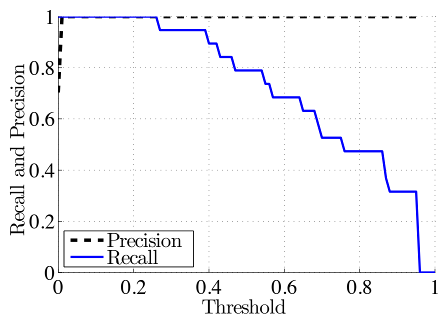


(a) 3 Transformations

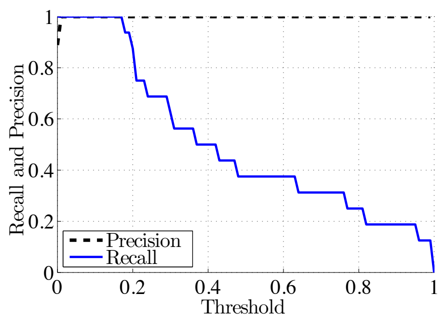


(b) 5 Transformations

Figure 8: Precision and recall of the copy detection system when videos are subjected to multiple transformations.



(a) View Synthesis



(b) Subset of Views

Figure 9: Precision and recall of the copy detection system when there are view synthesis and subset of views transformations.

4.3 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 28 query videos in Table 1 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. 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 0.3 and 0.8), the system yields 100% recall with more than 90% precision.

4.4 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 nine transformations described in Section 4.2 on all query videos. This means that we repeat the experiment nine times, 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 nine experiments are shown in Figure 7. 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., Gamma correction and texture blur, have more impact on the precision and recall values than other transformations such as logo insertion and flip.

We plot the average results across all nine transformations in Figure 6, 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.

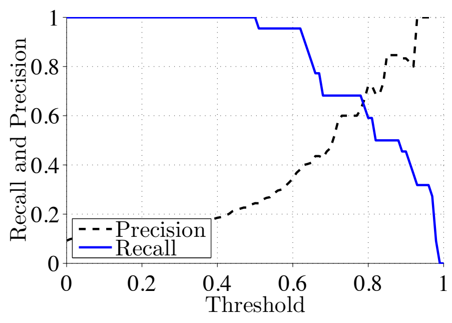
In summary, the results in this section show that the proposed system yields high precision and recall values in presence of various video transformations.

4.5 Multiple Transformations

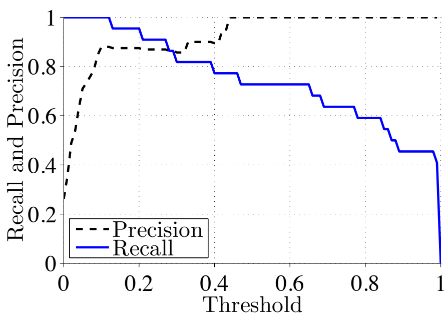
We assess the performance of our system in quite challenging scenarios, where each query is subjected to multiple transformations at the same time. We experiment with two cases. First, when each video is subjected to 3 different transformations, one applied on texture, one on depth, and one on both. 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 8(a) and 8(b). The results in Figure 8(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 75%. These experiments demonstrate the robustness of the proposed system to multiple video transformations.

4.6 View Synthesis and Subset of Views

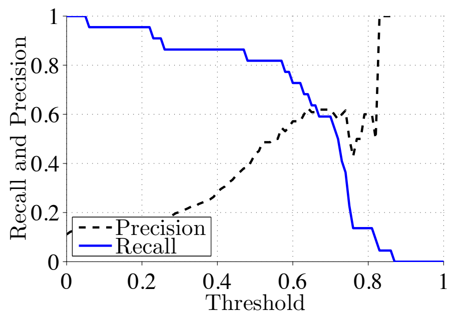
In these experiments, we apply two new transformations that are expected to appear in 3D video copy detection systems. The first one is called view synthesis, in which a virtual view is created from other views using software tools such as [4]. This is a challenging case to detect, as the synthesized 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 9(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 9(b). The results demonstrate the robustness of our system against the subset of views transformation.



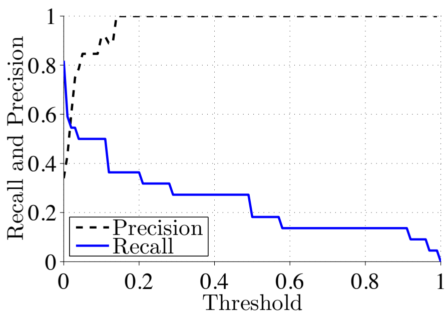
(a) Depth only, 3 transformations



(b) Visual only, 3 transformations



(c) Depth only, 5 transformations



(d) Visual only, 5 transformations

Figure 10: Precision and recall of the copy detection system when using depth and visual signatures separately.

4.7 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 4.5 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 10, which should be compared to the results in Figures 8(a) and 8(b) where both the depth and visual signatures are used in the detection process. The results show that there is a significant loss in the recall and precision values when we use either the depth signature or visual signature. Therefore, it is important to use both signatures to ensure good performance.

5. 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 284 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.

Acknowledgments

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada and the British Columbia Innovation Council (BCIC).

6. REFERENCES

[1] FFmpeg. http://www.ffmpeg.org/.

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

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

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

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

[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] 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.

[8] 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.

[9] 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.

[10] P. Kauff, N. Atzpadin, C. Fehn, M. MÃijller, 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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

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

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

[19] V. Ramachandra, M. Zwicker, and T. Nguyen. 3D video fingerprinting. In *Proc. 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV'08)*, pages 81 –84, Istanbul, Turkey, May 2008.

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

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

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

[23] 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.

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

[25] K. Tasdemir and A. 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.

[26] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

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

[28] 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.

[29] 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.