University of Edinburgh
School of Informatics

Detection of Deformed Video Copies

Undergraduate Dissertation
Artificial Intelligence and Software Engineering

James Anderson

April 1, 2009

Abstract: This report details the motivation and investigation of video copy detection (VCD). VCD aims to provide a method of comparing videos based on their frame content. We evaluated the performance of UQLIPS, an existing VCD system, when faced with various deformations and found that this decreased the accuracy of the results. BCS, the algorithm used in UQLIPS, was implemented and extended and it was found that splitting the frames into quadrants had the best performance, improving accuracy from 57.26% to 73.07%.

Acknowledgements

I would like to thank my supervisor, Prof. Bob Fisher, for his continued support, advice, and work over the course of this project.

I would also like to thank Shen et. al. for giving me access to the UQLIPS system and data, which aided the work and evaluation of this project.

Contents

1 Introduction 1
- 1.1 Project Overview . . . 2
- 1.2 Report Structure . . . 3

2 Background Research 5
- 2.1 Image Features . . . 5
  - 2.1.1 Scale-Invariant Feature Transform . . . 5
  - 2.1.2 Colour Histogram . . . 6
  - 2.1.3 Viewpoint Invariant Histogram . . . 6
- 2.2 VCD Algorithms . . . 7
  - 2.2.1 Bounded Coordinate System . . . 7
  - 2.2.2 Video Triplet . . . 7
  - 2.2.3 Dynamic Time Warping . . . 8
  - 2.2.4 Spatiotemporal Sequence Matching . . . 9
  - 2.2.5 Visual Indexing using Text Retrieval Methods . . . 9
  - 2.2.6 Video Sequence Symbolisation . . . 10
- 2.3 Summary . . . 10
  - 2.3.1 UQLIPS . . . 11

3 Performance of UQLIPS with Transformed Videos 13
- 3.1 Test Data . . . 13
- 3.2 Video Deformations . . . 13
  - 3.2.1 Translation . . . 14
  - 3.2.2 Rotation . . . 14
  - 3.2.3 Scaling . . . 15
  - 3.2.4 Skew . . . 16
  - 3.2.5 Noise . . . 16
  - 3.2.6 Contrast and Brightness . . . 16
  - 3.2.7 Merging . . . 17
- 3.3 Test Results . . . 19
  - 3.3.1 Evaluation Metrics . . . 19
  - 3.3.2 Results . . . 21

4 BCS Algorithm Implementation 25
- 4.1 Feature Extraction . . . 25
- 4.2 BCS Video Descriptor . . . 25
  - 4.2.1 Principal Component Analysis . . . 26
- 4.3 Comparison Algorithm . . . 28
- 4.4 Database . . . 28

4.5 Testing 30

5 New Feature Development **33**
5.1 Increased Length Histogram 33
5.2 HSV Colour Space 33
5.3 Viewpoint Invariant Histogram 34
5.4 Split Image 34
5.5 Colour Exclusion 35
5.6 Weighted Histogram 35

6 Results **37**
6.1 Extended Histogram 37
6.2 HSV Colour Space 38
6.3 Viewpoint Invariant Histogram 40
6.4 Split Image 41
6.5 Colour Exclusion 44
6.6 Weighted Histogram 47
6.7 Summary 48

7 Discussion **51**

8 Conclusion **53**
8.1 Future Work 54

Bibliography **55**

A Test Video Overview **57**

B Histogram Calculation Algorithm **59**

C Weighted Histogram Calculation Algorithm **61**

D RGB to HSV Conversion Calculation **63**

E Further results **65**

1. Introduction

In recent years, the cost of both digital cameras and storage media has decreased. Combining this with the increase in computer performance and compression standards, sharing multimedia information has become much more widespread. This is particularly prominent in video as the ever-increasing bandwidth of the internet has only furthered its exchange.

Websites like YouTube have facilitated the exchange of video, allowing users to upload their video with tags to summarise the content. This provides a method for other users to search for videos they are interested in. YouTube also recommends related videos based on these tags.

This creates some problems brought on by the nature of the system. The tags can only represent a general overview of the video content and the choice of tags themselves are subject to the user that uploads the video. Grouping videos based on their content would dramatically increase the usability of a video sharing system. This can be done using video copy detection.

Video copy detection (VCD) is based on methods of effectively indexing videos so that they can be compared based on their content. Most methods start by analysing the content of a video and use the features in the frames as a description for the video or subsection of the video (a scene, shot or group of visually similar frames)[13, 12, 9].

The focus of this report is on near-copy detection. The development of VCD has concentrated on two main areas: full- and near-copy detection. Both of these deal with full frame information; searching for copies of videos or partial matches of videos based on the entire frame as opposed to searching for videos within videos (a video featuring a TV in the background, for example). Full-copy detection searches for the exact copy, frame by frame for the whole video. Near-copy detection goes further than this to find videos that also share similar qualities (colour palette, people in the video or the location or event the video was filmed at). It also allows for videos where the sequence of frames do not match. For example, if they have been edited so that scenes appear in a different order (or excluded). Near copy detection is the focus of this report.

The motivation for this technique is clear: with the multitude of videos now available online, duplicates are inevitable. Duplication of copyright material is one of the driving forces behind this research. The legal ramifications of providing a video hosting service for public use is that the users will upload material that is not theirs. Companies want to protect their work and so they take legal action against the the host for "allowing" their videos to be uploaded into the system.

If duplicates could be reliably detected, then the videos that infringe copyright could be removed or redirected to point to the officially sanctioned copy provided by the creator.

Another benefit of VCD is the ability to remove duplicate videos. Although storage space is much cheaper, it is, nevertheless, more efficient to conserve space to store unique videos, allowing for more original content to be uploaded. Further to this, video quality can be improved by analysing any matched videos and referencing the highest quality (usually the video with the highest resolution) as the best result.

One of the more obvious benefits is simply the improvement of results. With a conventional search, any number of copies can only be returned by matching their tags with the search terms. Grouping these by video content itself not only would produce more relevant results but it would also ensure that each result is a unique video.

VCD research strives to meet three main aims:

High Recall All videos that match the query should be returned.

High Precision All videos that are returned should be a match to the query.

Efficiency Search time should be fast so that large databases can be searched quickly. Indexing these videos is not under the same constraints but ideally it should be possible in a short time.

Methods based on histograms [13, 12] and features [18] have all been investigated with respect to these aims. What is not clear is how these methods fair against different transformations that are common in edited video (colour changes, post-production effects like the addition or subtraction of elements within the video) and in video that is recorded from movies or at events (where the focus may be translated, rotated or skewed). This is the main area that will be investigated in this report.

1.1 Project Overview

Most of the current work assesses the performance of the algorithms in terms of precision, recall, efficiency and speed on untransformed videos and near-copies. What has not been tested extensively is their ability to cope under a variety of deformations typical of edited video. It is this area that is the focus of research for this project as a reliable VCD system should be able to locate copies even with transformations that alter the content of the video.

Prior research found the Bounded Coordinate System (BCS) [13] to be particularly accurate and efficient at comparing two videos. This representation is used for this project and it is the basis for the work carried out.

The main hypothesis of this project is:

> *Using a different feature descriptor will be more successful when used with BCS algorithm to match deformed video copies as opposed to the 64-dimension colour histogram currently used.*

This investigation only considers the alteration and subsequent identification of visual data. Audio, while a considerable component of most videos, is not assessed.

1.2 Report Structure

Section 2 details the prior research on other VCD systems and techniques. It covers the techniques examined and their strengths and weaknesses. Section 3 discusses the evaluation performed on UQLIPS, a website implementing video copy detection. It covers the deformations created and the motivation behind them. It also discusses the results of these tests and discusses possible improvements. Section 4 details the implementation of the BCS algorithm and the initial validity testing against UQLIPS. Section 5 furthers the work from section 4 by reporting suggested alterations to the algorithm to overcome the pitfalls in UQLIPS.

Section 6 explains improvements to the features that will be investigated and the aim of each. The results are given in section 7. Section 8 discusses the overall results of this testing and covers the improvement and extensions that could be employed to improve upon them.

2. Background Research

Most VCD algorithms rely on the use of features taken from the video's frames. Using these features, the algorithm builds a representation (or multiple representations) of the video. The algorithm also provides the ability to compare the representations to give a measure of similarity. With this, videos can be ranked according to how similar they are to the query.

This section first discusses common image features that can be used in these algorithms and then it discusses the algorithms themselves.

2.1 Image Features

Video copy detection algorithms rely on descriptions of the video content. Descriptions can be generated from all the frames in the video or just a selected subset.

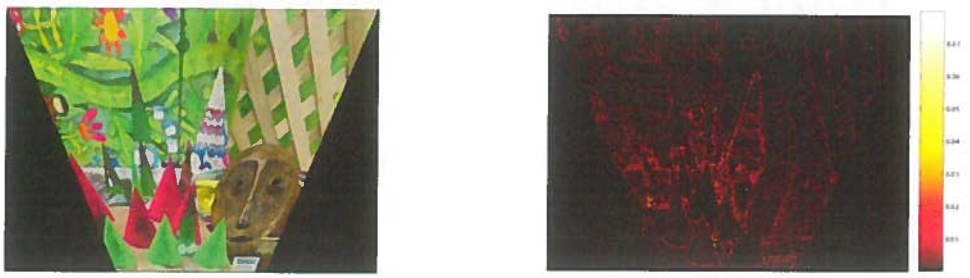
The features can be local, like SIFT (section 3.1.1), or global, with histograms of various qualities (sections 3.1.2, 3.1.3).

2.1.1 Scale-Invariant Feature Transform

The scale-invariant feature transform (SIFT) [11] is an algorithm that is used mainly for object recognition. These are local features for a given image and are robust against scale and rotation. They can also handle changes in illumination, noise and viewpoint (to an small extent).

A SIFT feature consists of a keypoint and the histogram neighbourhood to produce a 128-dimension feature vector. This is then matched by comparing it to other features in the database to locate matches. Many of these are used for objects within an image but more would be required to describe the entire image.

SIFT has been used in VCD algorithms [18] by taking key frames of a video and matching those with others to get 100% accuracy rates by using a high resolution frame but computing the similarity takes 1.2 seconds on average. This highlights the drawback of using SIFT for video as it produces a lot of data per frame which has to be matched within a database of hundreds of videos each with a large number of SIFT frames.



(a) (b)

Figure 2.1: Example image (a) and its weights (b)

2.1.2 Colour Histogram

A colour histogram is formed by binning pixels according to their colour values. Typically, this is done in RGB with a 3D description consisting of bins for the red, green and blue colour values. The normal method is to split the colour range of 0 to 255 into 4 bins from 0-63, 64-127, 128-192, 193-255. Doing this for each colour creates a 64-dimension feature vector from the possible bins (a bin being, for example, where one pixel contains red, green and blue values below 63).

This technique produces a general description of the frame (or all frames) and is quite fast to compute. It is robust to small changes in rotation and translation but any transformation that loses pixels or alters their values will impact the resulting histogram.

2.1.3 Viewpoint Invariant Histogram

A viewpoint invariant histogram [4] is similar to a colour histogram but it uses the gradients in the colour channels to apply weightings to the pixels. This allows for the image to be skewed or sheered without a loss of information as the transformation changes both the image and its weights equally.

This has not yet been used in VCD but it could provide a robust histogram that VCD algorithms can take advantage of. Figure 2.1 shows an example of how an image is weighted with respect to the colour gradients.

2.2 VCD Algorithms

Algorithms for VCD involve analysis of the content of each frame or key-frames of the video to measure how similar they are to another video. In this section, the different techniques for this process are discussed.

2.2.1 Bounded Coordinate System

The bounded coordinate system (BCS) [13] creates a global representation of a video. It uses frame features to summarise the general content in a compact representation. The video, $X = x_1, x_2, ..., x_n$ consists of n frames, each with a d-dimension feature vector (e.g. from a histogram of the frame). This produces the representation:

$$BCS(X) = (O, \ddot{\Phi}_1, \ddot{\Phi}_2, ..., \ddot{\Phi}_d)$$

Where O is the mean feature for all x, and $\ddot{\Phi}_s (1 < s < d)$ is an orientation and range d-dimensional vector. The orientation is a vector that represents the direction of the data with the greatest variance and the range is a value describing the extent of the data in that direction.

Comparisons can be executed in linear time by the following operation:

$$BCS(X) - BCS(Y) = |O_x - O_y| + \sum_{i=0}^{d} |\ddot{\Phi}_{xi} - \ddot{\Phi}_{yi}|$$

UQLIPS [13] is an implementation of this system and it produces, on average, a precision of 0.9 and recall of 0.6.

2.2.2 Video Triplet

The Video Triplet (ViTri) [12] representation represents clusters by hyperspheres. Using a descriptor for each frame (i.e. histogram), clusters are generated by the k-means algorithm and ViTri models a cluster, C, in the following way:

$$ViTri(C) = (O, R, D)$$

O is the origin of the cluster, calculated by taking the mean of all the data in the cluster. R is its radius, given by the standard deviation of the data in the cluster and D is the density, calculated by

$$D = \frac{|C|}{V(O,R)}$$

Where $|C|$ is the size of the cluster and $V(O,R)$ is the volume of the hypersphere with the origin O and radius R.

Comparisons are performed by searching for intersecting clusters. Clusters that don't intersect have no similarity and the ones that do are weighted by the density of the smaller cluster (to prevent similarity bias of large clusters). The system developed in [13] gives a precision and recall both, on average, 0.6.

This result is not as good as BCS (also compared in the paper, achieving a precision of 0.9 and recall of 0.6 on average) but the use of frame clustering with a density measure is quite novel.

2.2.3 Dynamic Time Warping

Dynamic Time Warping (DTW) [3] has been used as a distance metric for video clips to compensate for changes in order and framerate. It works by comparing two sequences and creating a map of the distances between two points. Algorithm 2.2.3 describes this method.

```
function DTW(sequence1, sequence2)
  int n = length(sequence1);
  int m = length(sequence2);
  int[][] dtw = int[n][m];

  for i = 1 to n
    dtw[0][i] = infinity;
  for i = 1 to m
    dtw[i][0] = infinity;
  dtw[0][0] = 0

  for i = 1 to n
    for j = 1 to m
      cost = distance(sequence1[i], sequence2[j])
      dtw[i][j] = cost + min(dtw[i-1][j],    // insertion
                             dtw[i][j-1],    // deletion
                             dtw[i-1][j-1]); // match

  return dtw[n][m];
```

Algorithm 1: Dynamic Time Warping Algorithm

This computes subsequence distances of each sequence so that `dtw[x][y]` is the minimum distance from `1` to `x` in `sequence1` and `1` to `y` in `sequence2`. In this algorithm, distances would be calculated between frames represented by histograms, features or pixel-by-pixel values. The returned value from this function is the distance of the two sequences which accounts for changes in framerate and inserted or deleted frames.

The system developed in [3] tested their implementation on a set of 712 videos and 8 copies of this data was generated to test the effects of brightness, frame rate (increase and decrease), lowering resolution and a mix of these. This mix was a specific hybrid of halved resolution, 10 fps (down from 30 fps) and fast motion (2× speed).

Assessing the false positive and false negative rates, of these transformations, on average the system performed very well. The average false positive rate over all alterations was 0.075 while the average false negative rate was 0.15. This is a very good result for these deformations and it clearly demonstrates that DTW performed very well on alterations to frame rate and differences in frame ordering (as fast motion effectively removes every second frame).

2.2.4 Spatiotemporal Sequence Matching

Spatiotemporal sequence matching [9] is a method proposed that takes a frame and divides it into $m \times n$ blocks. The average value of each block is calculated and converted to a rank based on this value. This produces a signature for the frame which can be easily compared.

Frame comparison uses a trade-off between spatial and temporal matching. Spatial matching produces a dissimilarity by computing the average distance between a subset of frames where the subset size is fixed to the size of the query video. Temporal matching uses a distance metric similar to dynamic time warping by averaging penalties applied for differences between two neighbouring frames.

The system in [9] recommends an equal weight of the dissimilarities produced by the spatial and temporal matching which produces a precision of 0.35 and a recall of 0.82. This was chosen by the author's decision to prioritise recall of copies over the precision of the results returned.

2.2.5 Visual Indexing using Text Retrieval Methods

As the general problem of VCD boils down to that of a search problem, there have been analogies to a thoroughly researched medium: text [14, 7]. To do this,

frames are treated like words so that the techniques that are used in text search can be used.

The common weighting function for text documents is *term frequency – inverse document frequency* (TF-IDF). This weights the word based on its frequency within the document while dampening that by the frequency of the document in the collection. In this scenario, the word is a frame (or set of similar frames) and the documents are videos.

This approach was used to find objects within a scene but this approach may lend itself to finding copies of a video by weighting the frames based on their occurrence within the database.

2.2.6 Video Sequence Symbolisation

Video sequence matching [1] uses the frame information to organise the video into a sequence of symbols. The symbols are created based on the feature descriptions of a frame so that similar frames are mapped to the same symbol. The symbol dictionary is generated from the feature space so that every similar frame in every video is assigned the same symbol.

Once the video has been converted into a sequence of symbols, these can be compared using edit distance. The edit distance operates in a similar way to DTW in that the distance between positions i and j in sequences a and b is given by:

$$d[i][j] = min \begin{cases} d[i-1][j] + del(a[i]) \\ d[i-1][j-1] + sub(a[i], b[j]) \\ d[i][j-1] + insert(b[j]) \end{cases}$$

Where *del*, *sub* and *insert* generate a cost for deletion, substitution and insertion respectively. These functions can either return the same penalty for any input or they can be adapted to dynamically weight each symbol.

For example, if a video consisted of two scenes each with three frames, it could be represented by the sequence *aaabbb*. This sequence would then match *aaabbb* exactly or *aabb* with two deletions.

2.3 Summary

Many of the methods researched are more applicable to the problem of locating a specific object within a video and while they can do this with great accuracy, the

approach does not lend itself particularly well to the problem of near duplicate detection due to the high dimensionality of the algorithms.

BCS performs well on near duplicate detection while maintaining a high level of efficiency and reducing the features into a compact summary. Using a histogram generalises the data but the BCS representation maintains enough to match similar videos with a high level of precision.

2.3.1 UQLIPS

UQLIPS is a web-based system implementing the BCS algorithm. The BCS algorithm used a 64 dimension colour histogram for each frame and built a representation from it. The full details on how the histograms are built is unclear but the assumption was that each channel was split into four bins and the RGB combinations of those created the 64 dimensions.

UQLIPS currently indexes around 10000 videos and a search can be performed by uploading a video to the website or selecting a video in the database to locate any near duplicates. Each video in the database has a unique identifier and so a search result in this system can be verified by comparing the ID of the query to that of the result.

A dissimilarity measure is included with the results which gives some indication into how the system compares the videos although the underlying implementation is unknown.

3. Performance of UQLIPS with Transformed Videos

Access to the system produced by Shen et al. [13] was given so that full testing could be performed on their implementation to assess the ability of the BCS algorithm with regard to deformed videos. The system in question only indexes original, untransformed videos but a deformed video can be uploaded as a query. It is not added to the database when uploaded so testing can be carried out with the knowledge that there is only one match to the original video.

3.1 Test Data

Test data was taken from the UQLIPS database. It indexes around 10000 videos, mainly adverts and film trailers of approximately 1 minute in length, consisting of approximately 1500 frames.

The length of these videos is not particularly long but maintaining a fixed length ensures that any variation in results is due to the deformations themselves. The videos have an audio component but UQLIPS does not use this.

10 videos were taken at random from UQLIPS as an initial pool from which to create test data. 22 transformations were applied to these and to create 220 videos. This is a small set of data to use but due to the constaints on time and storage space, increasing this number was not feasible.

Appendix A shows an overview of the videos chosen.

3.2 Video Deformations

This section details the deformations chosen and the motivation behind them. They were restricted to the type of transformation usually found in edited video. Each deformation was tested with both a minor and a major transformation (which is discussed in each section as appropriate).

(a) (b)

Figure 3.1: Translation of 5 (a) and 50 (b) pixels

3.2.1 Translation

The translation transformation was a simple shift of the pixel data by a specified amount. Translation was chosen as it is quite common to find videos where the object being recorded is not in the centre of the frame. It is also a good test of the robustness of the algorithm to loss of pixel data as BCS operates on colour histograms so any change in pixel data is likely to propagate a change in the computed representation.

Translation was performed with a shift of 5 pixels on x and y from the top-left corner and with a shift of 50 pixels, shown in fig. 3.1.

3.2.2 Rotation

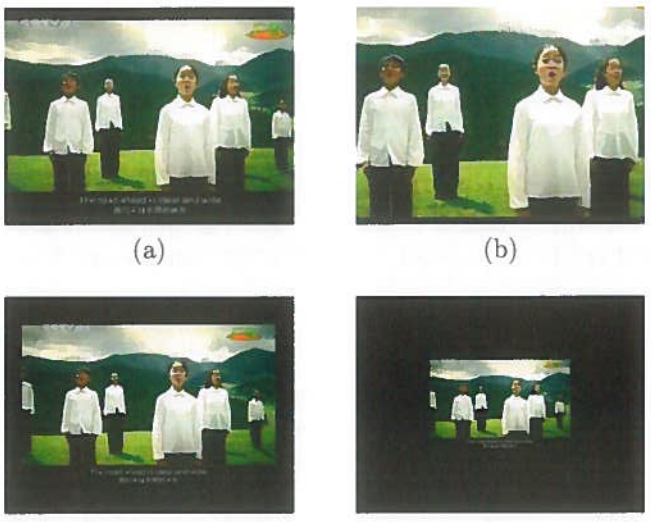
Rotation of a video in this case is a pure rotation around the centre. This alters the immediate neighbourhood of pixels and it creates a loss of pixels at the extremities of the image as they are rotated off of the visible plane. This is also a common deformation in videos either as a result of post-processing or simply a choice in the framing of a shot.

Similar to translation, this transformation changes some of the pixel data but to a lesser extent. Image histograms are normally quite robust to this type of transformation and so the impact on the results was expected to be minimal.

The video was rotated both 5° and 20° counter-clockwise as shown in fig. 3.2.

(a) (b)

Figure 3.2: Rotation of 5° (a) and 20° (b)



(a) (b)

(c) (d)

Figure 3.3: Scaling of 10% (a), 50% (b), -10% (c) and -50% (d)

3.2.3 Scaling

The next deformation is scaling. This is very common in recorded video where the centre or some other region of interest is the focus (enlargement) or the full content is captured along with an extraneous border (shrinking). Again, this changes the pixel information present and thus the histogram and accompanying BCS representation. In the case of video shrinking, the video not only suffers from a loss of information but there is increased noise from the the surrounding border of the object.

The video was scaled both up and down, producing an enlargement or shrinking, by 10% and 50%, shown in figure 3.3.

(a) (b)

Figure 3.4: Skew of 10° (a) and 45° (b)

3.2.4 Skew

If you picture the scene of a person recording a film in the cinema, quite often the resulting copy is viewed at an angle to the screen. While the majority of the content is recognisable, much the underlying data is missing. It also introduces the same border problem as scaling which further alters the histogram of the video.

The skew angle was calculated from the normal view of the video at 0° and this was rotated 10° and 45°. This is shown in the examples in fig. 3.4.

3.2.5 Noise

Before digital TV, noise was a common element in the signal. While it is not common in digital media, it can occur due to errors in capture (due to incorrect exposure) and it can be present in the medium being captured (e.g. photographic film or TV).

Another reason that this deformation was chosen was that it alters the histogram globally by changing random pixels to different values. Gaussian noise was used with a mean of 0 and a variance of 0.01 and 0.05 to produce noise like the examples show in in figure 3.5

3.2.6 Contrast and Brightness

Another common transformation to video is contrast and brightness alteration. These can be changed manually or they could arise as the result of the environ-

(a) (b)

Figure 3.5: Noise with variance 0.01 (a) and 0.05 (b)

ment where the video was recorded. These tests are particularly prominent as they alter the histogram on a global scale.

Brightness alteration has the effect of shifting the entire histogram up or down. In histogram distances measures, it can be classed as a large distance on any absolute subtractions but any measure that assesses the shape of the histogram should be robust to this.

Changes in contrast are slightly different to this. They squash or stretch the histogram in the range. This alters the shape but it is possible to compensate for this by first normalising the histogram.

Alterations to brightness and contrast are shown in 3.6 and 3.7, respectively.

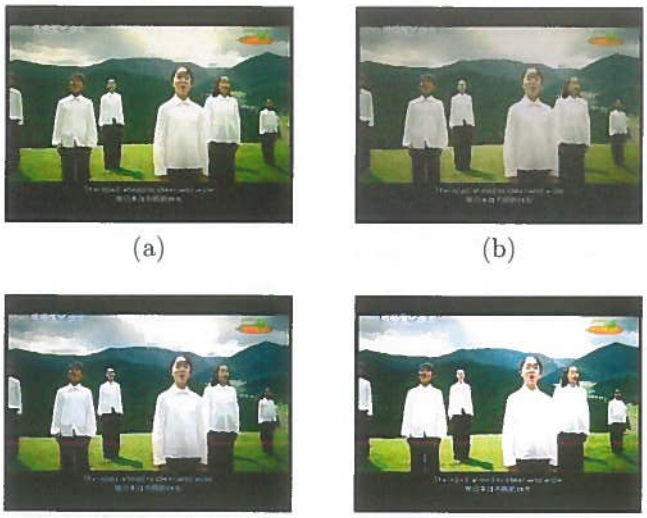
3.2.7 Merging

Another interesting transformation is the combination of frames from two different sources. This is common in edited video clips, films (e.g. trailers) and TV programmes (e.g. adverts). This result of matching this deformation against the original two videos will be interesting as it will highlight how well the BCS can summaries two videos in one. This is expected to be quite tough as the representation will not encompass the two videos but rather it will be placed in the mean of their features, creating a new representation quite different from either original video.

Merging two videos was performed by taking half of the frames in one video and half in another and joining them together to make a new video. Figure 3.8 shows the point where the videos are merged together, highlighting the hard cut between the frames (as they are not blended or faded like typical video transitions).

(a) (b) (c) (d)

Figure 3.6: Brightness decrease of 10% (a), 50% (b) and increase of 10% (c) and 50% (d)



(a) (b) (c) (d)

Figure 3.7: Contrast decrease of 10% (a), 50% (b) and increase of 10% (c) and 50% (d)

Figure 3.8: An example merged video, showing the point of data merge

3.3 Test Results

To test UQLIPS, every deformed video was uploaded through the web interface. This produced a a page with top 15 results of the query. Each result consisted of the ID of the video (with each video possessing a unique ID) and dissimilarity between the query and result.

Ideally, more than 15 results would be taken form the system but they could not be obtained as an attempt to access the next 15 results produced a server error.

3.3.1 Evaluation Metrics

Precision and recall are the typical measures for a search problem but as there is only one match to any transformed query, these measures become much less relevant.

For example, the precision (calculated by the equation below) of a video query would produce a result of $\frac{1}{15}$ if found in the results or 0 if not found. This is because there is only one possible match in the results for the query: the video from which that copy was made.

$$precision = \frac{number\ of\ copies\ of\ the\ query}{number\ of\ videos\ retrieved}$$

Similarly, recall (calculated using the formula shown below) would produce either a result of 1 or 0 as there is only one copy in the database which is either recalled or not.

$$recall = \frac{number\ of\ copies\ of\ the\ query}{number\ of\ copies\ in\ the\ database}$$

This section introduces the metrics that will be used to evaluate the results of both the testing of UQLIPS and the subsequent testing of the BCS implementation and feature experimentation.

3.3.1.1 Position Accuracy

To compensate for the lack of typical precision and recall measures, a metric was created to assess the ability of the algorithm based on the position of the result. This allows the result of a query to be assessed independently of the underlying BCS algorithm.

The position accuracy measurement applies a score to a matched result. This is calculated by:

$$Pacc(q) = 1 - \frac{Pos(Match_q) - 1}{number\ of\ results\ returned - 1}$$

Here, $Pos(Match_q)$ is the position where the copy of query q was found and *number of results returned* is how many results are taken from the query, for UQLIPS this is 15. It produces a score of 1 for a match in the first position and 0 for the last result.

Position accuracy places emphasis on where the match to the query is found, with the best possible being in the top position (and by implication, the result with the lowest dissimilarity). This was devised because for the tests executed on UQLIPS, there is only one match to a deformed video and it is important that it is accurately detected as a copy by the system.

3.3.1.2 Confidence

Investigating the dissimilarity measure of the results can give some insight into how the algorithm works on the data given and how the position of the results are generated. This inspired the creation of a *confidence* measure to decide if a match to a query is statistically significant compared to its neighbour. The value is calculated by the following formula:

$$C(q) = \frac{(\vec{diss}_{Pos(Match_q)+1} - \vec{diss}_{Pos(Match_q)}) - std(\vec{diss})}{3 * std(\vec{diss})}$$

$\vec{diss}$ is the dissimilarities for each position and $Pos(Match_q)$ is the position where the copy of query q. $std(\vec{diss})$ is simply the standard deviation of the dissimilarity vector. The metric is normalised to the range -1 to 1 by dividing by treble of the standard deviation. This captures 99.73% of the data correctly but there is a 0.27% chance that the confidence would be greater than 1. This was seen as an acceptable margin of error due to the likelihood of occurrence.

The reason this was not simply tested for the most similar result (in the first position) was that if the result was not the target for the query then the confidence measure is not as relevant. It may be an insight to see if the algorithm produces a high confidence for a non-match in the first position but it would only indicate that it is confident in finding the wrong video, not its confidence in comparing the correct copy.

This also allows for the case where the top result may be a similar copy of the video in the set but it is not the exact copy taken for this test (as there are multiple copies of the same video in the UQLIPS system).

While this metric will rarely produce values at the extremes, it allows for comparison between the UQLIPS system and the project implementation.

3.3.2 Results

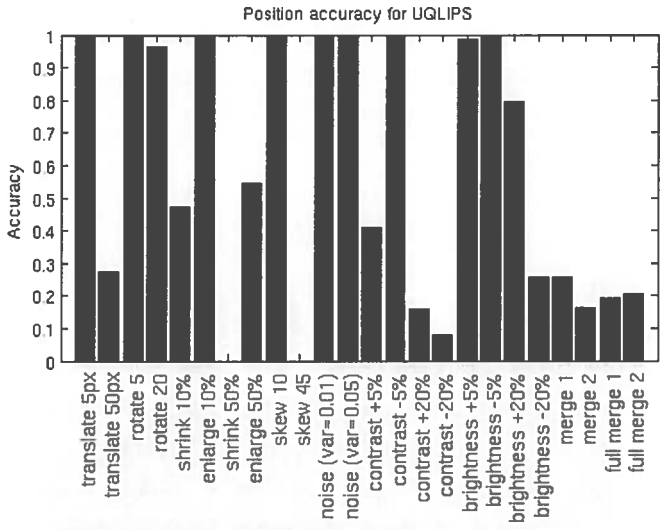


Figure 3.9: Results of testing each deformation using UQLIPS

After submitting each video to UQLIPS, the results in fig. 3.9 were obtained. Figure 3.9 shows the position accuracy of each deformation averaged over the 10 transformed videos.

Looking at the minor transformations, BCS is quite resilient and they remain unaltered. The exceptions to this are the effects of shrinking, increasing contrast and merging.

The reason that shrinking, even by 10%, has such a profound effect on the resulting position is not just that data is lost but that much more noise in the form of black pixels is introduced. As the shrinking operation wraps a black border around the original content, this too is taken into the algorithm where it will alter the resulting histogram produced.

The drop in accuracy for the small contrast increase can be explained if the histogram is considered. When the contrast is increased, the histogram for it is stretched across the range. This will alter the BCS by producing a result where

the origin changes and the ranges are reduced, creating a normalising effect. The orientation should remain mostly intact as the general trend of the video will be close to the original. In essence, this creates noise as the BCS becomes similar to other, unrelated BCSs.

Merged video caused BCS some trouble which was expected because of the very nature of representation. Taking a summary of a video is problematic when the video itself covers a lot of varying content. In this respect, mixing two videos together is not the same as mixing their summaries together because the summary is an average of their content which in turn does not match to either of the source video's summaries.

Along with these deformations, each major deformation was problematic for the BCS. The results were expected to be poor as the deformations were designed to cause significant changes to the content of the video.

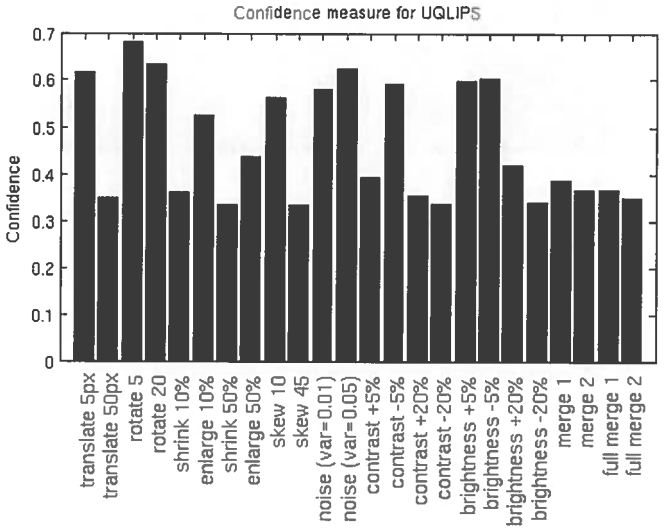


Figure 3.10: Confidence measure for each deformation using UQLIPS

Figure 3.10 shows the confidence for each deformation tested.

It can been seen in the figure that for the deformations which had the worst performance in figure 3.9, the algorithm had the lowest confidence. This means that when the results were found within the top 15, the associated dissimilarity was approximately the same as its neighbouring results (either those with lower dissimilarity or higher). This indicates that these transformations produced enough noise to cause the original video to be seen as similar to many other videos in the system.

These results show that while BCS is robust against small transformations to the

videos, it is adversely affected by the major transformations. The next section details the implementation of the BCS algorithm and the testing performed to assess its accuracy with respect to the UQLIPS implementation.

4. BCS Algorithm Implementation

The initial step for this project was to implement the BCS algorithm and ensure its accuracy. This was decomposed into three separate tasks: extracting the required features from the video, creating the BCS representation of the video using the features and finally, implementing the comparison algorithm to calculate the dissimilarity between two videos.

4.1 Feature Extraction

As this algorithm takes an input in the form of features, a function was created to extract these features from frames of a given video. Mirroring the UQLIPS system, the features took the form of a colour histogram. The exact nature of the histogram used in UQLIPS was unknown but as it was stated that it had 64 dimensions, the assumption was that it combined 4 bins for each colour channel ($4 \times 4 \times 4 = 64$).

The algorithm used to create this histogram is given in Appendix B.

4.2 BCS Video Descriptor

Creating the BCS representation from the video's features needed three separate components: the origin, the orientations and the ranges. These are all calculated from the data which takes the form of a matrix X. X is an $n \times d$ matrix where n is the number of frames in the video and d is the length of the histogram used to describe it.

The calculation method explained here is not explicitly stated in [13] as the method that was used, rather the BCS components (origin, orientations and ranges) were given and it was decided that this was the most accurate way to calculate the components. This section covers each component in turn, starting with the origin.

The origin was simply calculated as the mean of all the features:

$$O = \frac{\sum_{i=0}^{n} \vec{x}_i}{n}$$

Where n is the number of frames and $\vec{x}_i$ is the feature vector from the ith frame (the ith row of X).

The calculation of the orientations uses Principle Component Analysis [8] to analyse the data and find the directions in the data with the greatest variance.

4.2.1 Principal Component Analysis

This section covers the main steps of principal component analysis and its relation to the BCS representation. An example shown in figure 4.1 summarises this using a 2D example.

The first step in this process is to calculate the mean for each dimension (and as the origin is the mean, it can be reused in this step). The mean is then subtracted from the data to centre it around the origin.

To standardise the data based on the correlations, z-scores [10] are calculated by the following formula:

$$Z = \frac{X - \mu}{\sigma}$$

Where μ is the mean of the data, X and σ is the standard deviation.

By doing this, the data is normalised so that the resulting principal components are more comparable [8].

The eigenvectors and eigenvalues of the data are then calculated. The next subsection discusses the properties that define eigenvectors and eigenvalues and how they are calculated.

4.2.1.1 Eigenvectors and eigenvalues

An eigenvector is a vector v such that:

$$Tv = \lambda v$$

Where T is a square matrix and λ is the eigenvalue for the eigenvector. What this means is that the eigenvector does not change direction when multiplied by the data, it is only scaled by λ.

This can be done in matrix form and rearranging the sides gives the following:

$$V^{-1}TV = D$$

In this equation, V is the matrix of eigenvectors for each dimension and D is the diagonal matrix of eigenvalues where:

$$D(m,n) = \begin{cases} \lambda_i & \text{for } m = n = i \\ 0 & \text{for } m \neq n \end{cases}$$

To calculate these matrices we need to use a square matrix but this can also be achieved with non-square matrices by using Singular Value Decomposition (SVD) [17, 16].

4.2.1.2 Singular Value Decomposition

SVD is the factorisation of data $n \times d$ matrix, A, such that:

$$A = USV^T$$

Where U is an $n \times d$ matrix, S is the $d \times d$ diagonal matrix and V^T is of size $d \times d$. Both U and V are orthogonal so $U^T = U^{-1}$ (similarly for V).

S can be computed by calculating the eigenvalues of AA^T or A^TA (as they are both the same). This is done by solving the characteristic polynomial of AA^T given by the expansion of the determinant satisfying:

$$det(AA^T - \lambda I) = 0$$

where I is the identity matrix. Solving this equation gives the eigenvalues and taking their square roots and ordering them by decreasing size gives the diagonals of the matrix S.

Using this fact that V is orthogonal, it can be calculated as the eigenvectors of A^TA as shown below:

$$A^TA = (USV^T)^T(USV^T) = VSU^TUSV^T = VS^2V^T$$

These are the "right" eigenvectors (with U corresponding to the "left" eigenvectors). V and S correspond to the eigenvectors and eigenvalues of A and this technique is applied to the data matrix Z.

Using the eigenvalues from S and the eigenvectors from V, the PCA algorithm would continue by selecting a subset of the eigenvectors to describe the data. It would then adjust the data by multiplying it with the eigenvectors so that the points lie along the axis described by the vectors. This process is shown in the bottom left panel of fig. 4.1

The BCS algorithm uses the eigenvectors as the orientations for each dimension and the original data is not adjusted.

The range for each dimension was calculated by rotating the data so that it lies along the axes (the product of XV where V is the eigenvectors calculated by SVD. After the data has been rotated, the standard deviation for each dimension is calculated and doubled. The resulting vector captures the range of 95.45% of the datapoints, leaving only outliers outwith the bounds. An example of the bounds produced is shown in the bottom right panel of fig. 4.1.

4.3 Comparison Algorithm

To compare two BCS representations the following equation was used:

$$BCS(X) - BCS(Y) = \frac{\frac{|O_x - O_y|}{max(O_x, O_y)} + \frac{\sum_{i=0}^{n} |r_{xi} - r_{yi}|}{2d} + \frac{\sum_{i=0}^{n} |b_{xi} - b_{yi}|}{max(sum(\vec{b_x}), sum(\vec{b_y}))}}{3}$$

This sums the difference of the origins, of the range for each feature and of the bounds of each feature. Each component is normalised between 0 and 1 by division by each components maxima and then finally dividing by 3 to average the differences.

The subtractions account for differences in translation (from the origin), orientation (from the ranges) and scale (from the bounds).

4.4 Database

To properly test our implementation of the BCS algorithm, a sizeable set of videos should be downloaded so as to closely match the UQLIPS environment. UQLIPS

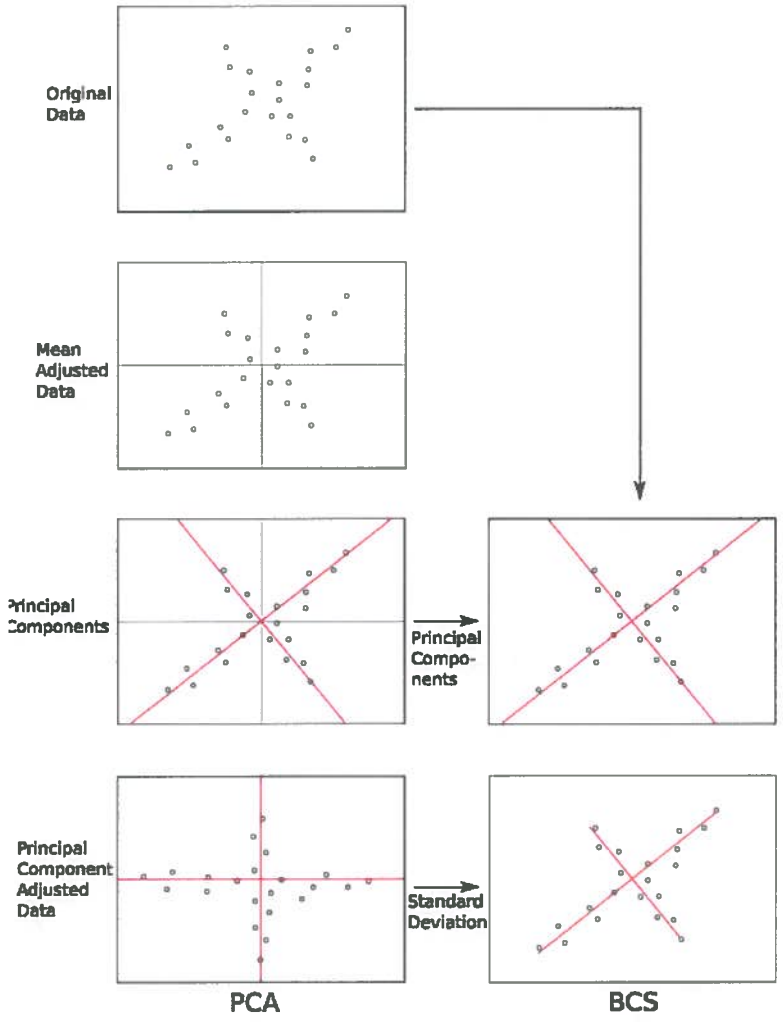


Figure 4.1: An example of how PCA is used in BCS

has a database of around 10000 videos but for the purposes of this project, that number could not be matched. There were two reasons behind this: time and space.

The project had a strict time constraint and as it takes time to generate the features for each video, using too many would be very costly in the testing phase.

There is also concern for the storage space required by these videos. An individual video varies from 6Mb to 10Mb but storing thousands along with their extracted features would use more space than there was available.

To solve this problem, the results from the testing of UQLIPS were analysed and all the videos that were found in the top 15 results for each query were downloaded. The reason this subset was taken was to extensively test the implementation as it was known that the videos contained in the results had comparable or lower dissimilarity for the deformed video query. In this way, if a query in our implementation is found in a lower position (e.g. 2nd instead of 3rd) compared with the UQLIPS result then it must be because an improvement was made.

For example, a query, def, was tested on UQLIPS to give the results x, $orig$, y, etc (where $orig$ is the desired match for the deformation, def). Then, with the same three results present in our implementation, def was used as a query and gets the result $orig$, z, x, y etc. This is an improvement over the UQLIPS implementation and although the results differed, this must be an effect of the new features used in the implementation. It gives a different dissimilarity (in favour of the original video), but it would still compare def to x and y even though the are present in a different order.

The only situation where this would not work is if the video dissimilarities were completely altered by one of the experiments. This could then rate the videos in our subset as less similar than the original while rating videos not present as more similar to the video. If this was the case then it would affect all results for that experiment significantly but unless that is the case, it is assumed that the subset obtained provides an equal challenge to that of the UQLIPS implementation.

After the videos from the results were taken, a further 100 were also added to produce a subset of 517 videos.

4.5 Testing

To ensure that the algorithm was implemented correctly, every original video was supplied as a query of a search on all videos in the database (query included). This produced a correct match in the first position and with 0 dissimilarity 100% of the time.

The deformed videos were also tested against the set of original, untransformed videos (no other deformed videos were included in the search). This produced the results show in figure 4.2.

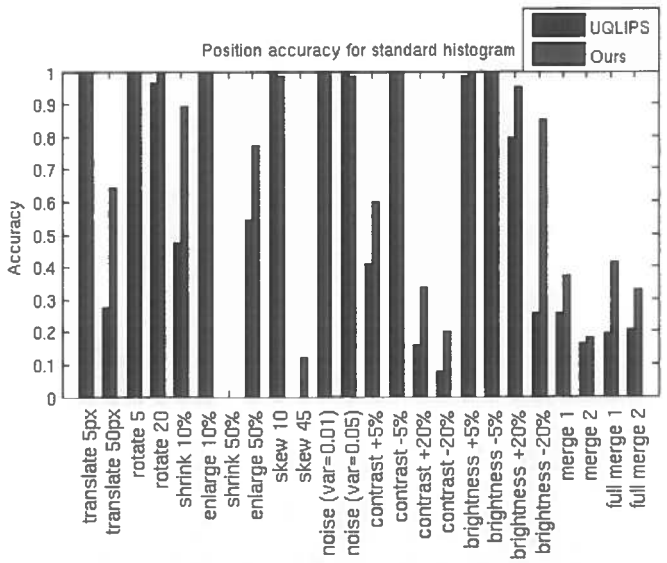


Figure 4.2: Summary of results of testing each deformation using the basic implementation

What is quite interesting is that for most of the deformations, the implementation performed better than UQLIPS. This can be explained by a difference in implementation. The calculations used in UQLIPS are unknown and the method used in our implementation was chosen because it was thought it would represent the data accurately.

The representation may also be a factor. The orientations were stored as a rotation matrix but is not the only way to represent a rotation. As such, the BCS representation used in UQLIPS may calculate a slightly different orientation to the one calculated in ours.

As the difference has only improved results, this does not pose a problem that prevents further exploration for the continued improvement of the results.

Figure 4.3 shows the confidence of our implementation compared to UQLIPS. The improvement is not only reassuring but it provides some evidence that the accuracy improvement previously explained was not due to increased noise in the dissimilarity measure. What this means is that for each improved result, not only was the accuracy higher but the difference between the dissimilarities of the result and its neighbour has increased (highlighted by the increased confidence).

Overall, the accuracy improved from the average 57.26% from UQLIPS to 69.23% along with a confidence boost from 46.4% to 49.77%.

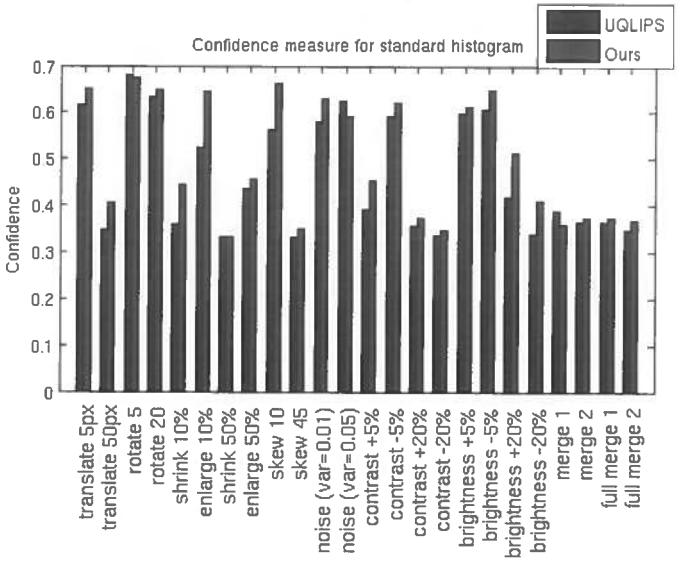


Figure 4.3: Summary of the confidence each deformation using the basic implementation

5. New Feature Development

As the results from UQLIPS showed, the deformed videos have quite an effect on the results returned. BCS builds a representation from the features used and the video is summarised by the general trend of those features and the implication of this is that if the features confer some robustness to a deformation then the BCS produced should also possess the same level of tolerance.

This section covers the altered features tested and the motivations for choosing them. The features either aim to increase accuracy for the structural deformations (translation, rotation, skew and scale) or colour deformations (noise, contrast and brightness). There are no specific features that target the merged videos but it is of interest to see if they can be improved with a different global representation.

5.1 Increased Length Histogram

The histogram used in UQLIPS separates each channel into 4 bins, each covering 64 colour values, producing a 64-dimension vector. This is quite a coarse summary of the frame and so the initial idea was to increase the bins to 5 so that the bins cover 51 colour values and produce a 125-dimension vector.

This was expected to increase results for all deformations as it build a larger, more detailed summary of the video content.

5.2 HSV Colour Space

The RGB colour space directly represents the colour value but Hue-Saturation-Value [15] colour space represents these in a different way. The hue is a representation of the colour itself, saturation is how colourful the colour is (with 0 saturation creating grey), and value represents how light the colour is.

It was hoped that by using this colour space it will be tolerant to changes to brightness and and contrast as they alter only one (brightness alters saturation) or two (contrast alters saturation and value) rather than all three in RGB.

The calculation to convert RGB to HSV is given in Appendix C.

(a) (b) (c)

Figure 5.1: Horizontal split (a), vertical split (b) and quadrant (c)

5.3 Viewpoint Invariant Histogram

In the initial research, the viewpoint invariant histogram captured image content by weighting the pixels with respect to the colour gradients. This created a histogram that was robust to changes in viewpoint and by using it in the BCS descriptor it was hoped that it would improve performance of the skew deformation.

The author of [4] also release a library for the computation of this histogram which was used to create them from video frames.

5.4 Split Image

While one summary for a video provides an adequate representation for the content, splitting the content and building separate summaries can take advantage of the natural composition of video. In particular, there is often a split between the top and bottom areas of a scene (e.g. ground and sky).

Using this knowledge, the frames were split into two histograms: one that was created from the top content of the frame and one that used the bottom content. This was extended to produce two more sets: a split of the left and right sides and a quadrant split. These are shown in fig. 5.1.

These are stored as 2 (or 4) BCS representations for each video. Although this doubles (or quadruples) the data to compare, it is the only way to isolate the data for each section. When comparing the videos, the representations are compared top to top, bottom to bottom etc. and the average dissimilarity is returned as the overall dissimilarity.

The three variations are not targeted to a specific deformation but are more likely to increase the accuracy of the structural deformations as opposed to the colour deformations.

Figure 5.2: Weighting of pixel values, doubling in the centre circle

5.5 Colour Exclusion

One of the main problems that the deformations possessed was that they contained a lot of noise due to the addition of pixels with no relation to the content. This is particularly evident in both shrunken and skewed videos where the area around the video contained black pixels where there was no content.

The idea behind this experiment is that the noise can be reduced if those pixels are not counted. The simple way to do this is to exclude any pure black pixels (RGB values are all 0). This was mirrored with pure white pixels as both of these are least likely to occur naturally in a video. This is also hoped to improve the result for deformations that increase brightness and contrast.

5.6 Weighted Histogram

Taking into account that the focus of a video (or copy of a video) is normally the centre of the frame, weighting the values in the centre of a frame should increase their prominence in the representation. This should counteract the effects of scaling, rotation and skew as the noise introduces at the edges of the frame should not affect the representation as much as the higher weighted centre.

The operation of this is shown in 5.2. The weight mask is centred in the frame with a radius given by $\frac{min(w,h)}{3}$ where w and h are the width and height of the frame, respectively.

The exact calculation algorithm is shown in Appendix D.

6. Results

This chapter covers the results for each feature tested. All results are plotted alongside the results of our implementation using the standard size histogram so they can be easily compared.

6.1 Extended Histogram

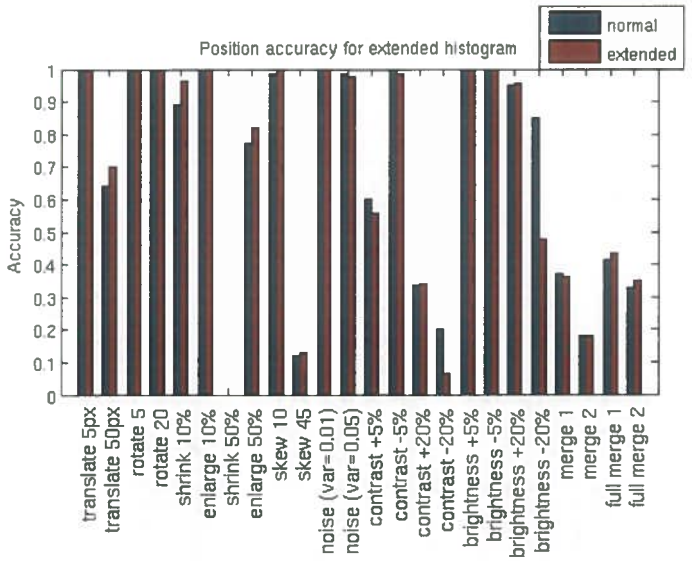


Figure 6.1: Results of testing each deformation using extended histogram

The extended histogram was expected to perform better across all deformations as it effectively increases the feature space. By doing this, each video would have a more comprehensive summary to use in comparisons.

The results in figures 6.1 and 6.2 show that this theory is only half true: it does increase the performance on the translation (50px), shinking (10%), enlargement (50%), noise (0.05 variance) and skew deformations. Contrast and brightness show a large drop in their major forms (-20%). This is most likely because these reduce the difference in the colours of the frames and thus produce a more general BCS. The confidences both drop for these deformations which solidifies this reason as it shows that the result was more random than a confident match.

Reviewing the confidence on the deformations that were found more accurately also shows a drop which indicates that these may only have improved due to chance rather than the effect of more information in the histogram.

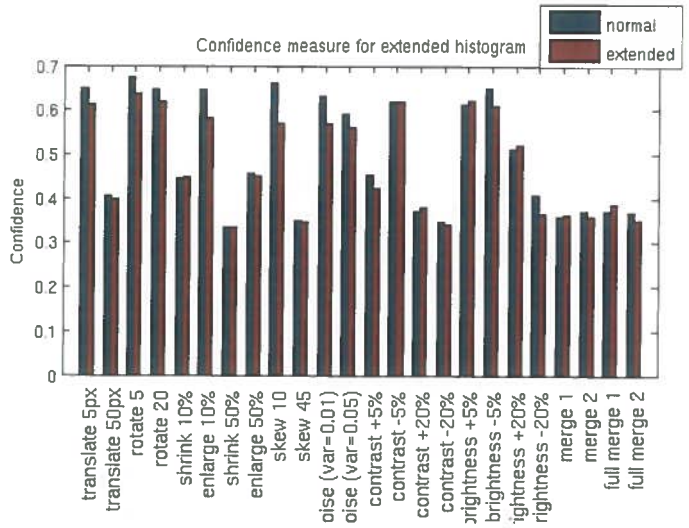


Figure 6.2: Confidence each deformation using extended histogram

6.2 HSV Colour Space

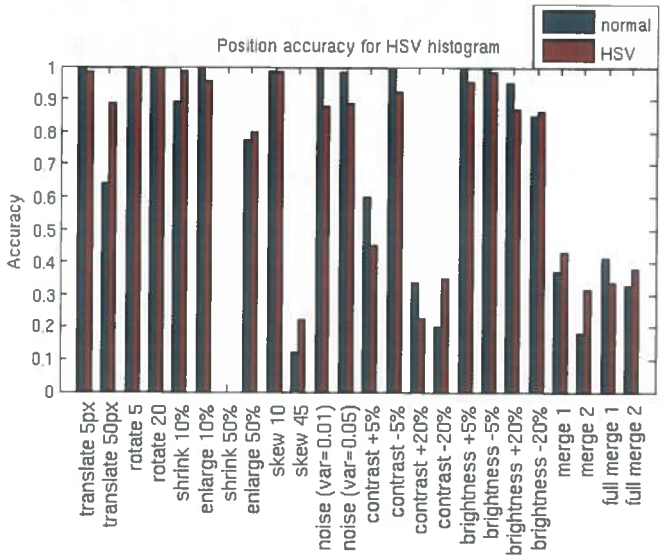


Figure 6.3: Results of testing each deformation using HSV colour space

Using HSV colour space was meant to improve the ability for the algorithm to match copies with altered brightness and contrast as it represents these alterations differently to RGB.

The results in fig. 6.3 show that it does not really have this effect. The confidences are shown in fig. 6.4. Although it improved the cases where contrast and

brightness were reduced by 20%, it performs worse for the other levels of alteration. The improvement on the two reduced deformations is quite interesting as they performed the worst when using the normal histogram. The use of HSV also increased the confidence of the brightness -20% deformation which shows that it was not simply improved by chance. However, the confidence of the other brightness and contrast deformations (with the exception of a brightness increase by 20%) is decreased, but for contrast it is a minor drop compared with brightness. The increase for a brightness increase by 20% is interesting as it implies less noise although the results were worse. However, the confidence is quite low compared with deformations with better accuracy and so it is thought that the results still contain noise.

HSV also appears to provide some robustness to the introduction of black pixels caused by translation, shrinking and skew. This is backed up by their confidence levels and the reason for this may be that the remaining content pixels form a representation that is more similar to the original than RGB values. This can also apply to the merged videos as they contain the same content (no colour alterations) and their accuracy increased along with a slight confidence increase on average.

As the representation appears to match better when the colour range remains the same, this explains the drop in accuracy for the noise deformations (where the colours are shifted randomly). Accompanied with a large confidence drop, HSV appears much less tolerant to changes in colour values than RGB.

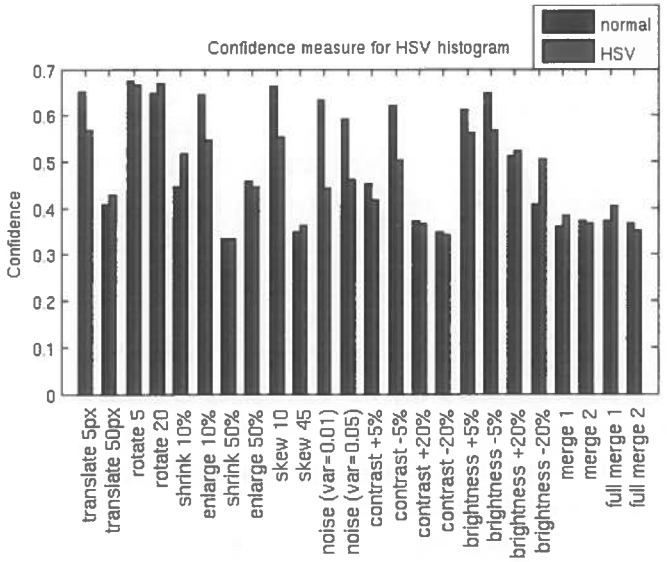


Figure 6.4: Confidence each deformation using HSV colour space

6.3 Viewpoint Invariant Histogram

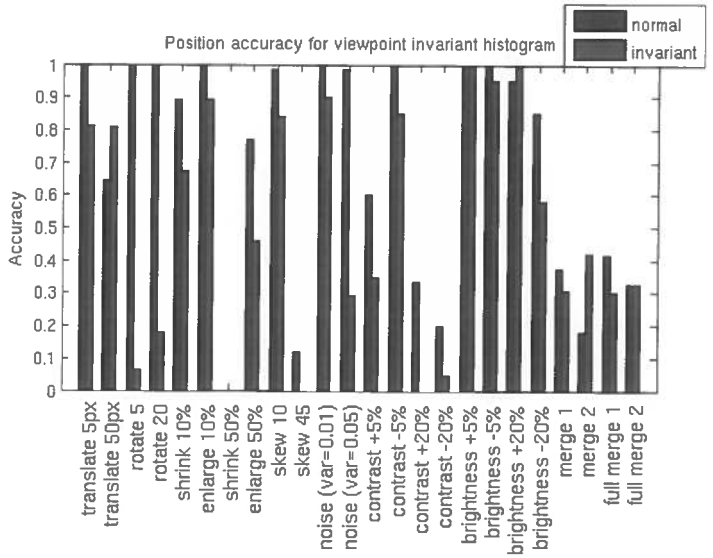


Figure 6.5: Results of testing each deformation using viewpoint invariant histogram

The viewpoint invariant histogram created a histogram that was meant to match a skewed image closely to its original counterpart by weighting the pixels based on the colour gradients. Figure 6.5 shows the results and figure 6.6 shows the confidence.

The result of its application to the BCS representation dramatically reduced accuracy and confidence for almost all deformations, including the skewed videos. The reason behind this is that the histogram produced is scaled down by the weights applied. This moves the mean closer to the origin and introduces much more noise which in turn lowers the accuracy for the deformations.

This idea is backed up by the almost complete drop in confidences for all deformations. The only improvement, both in confidence and accuracy was for the major translation and brightness operations. The reason behind the translation increase is the loss of data which makes the representation much more likely to match to videos that show a similar trend even though the translated video will only match partially (as a lot of data is lost). Brightness increased because it has the effect of increasing the difference between colours, which would increase the weights in the histogram, making it more unique and thus, less similar to most other videos (it would, of course, still be similar to the original).

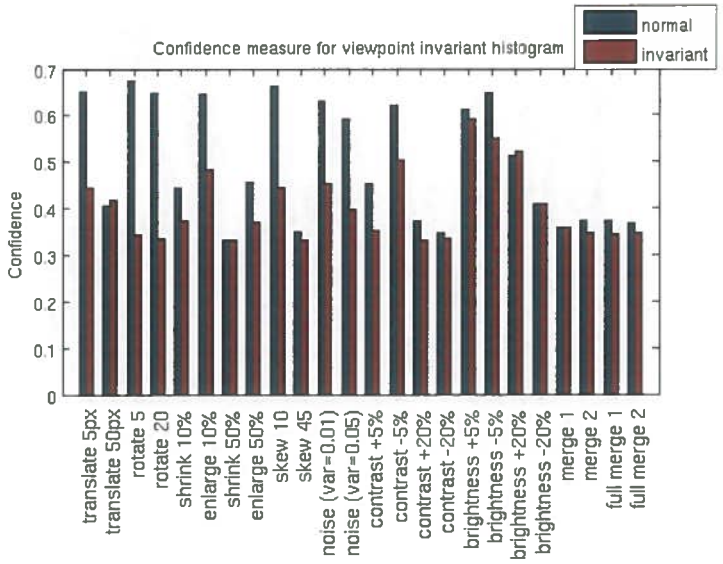


Figure 6.6: Confidence each deformation using viewpoint invariant histogram

6.4 Split Image

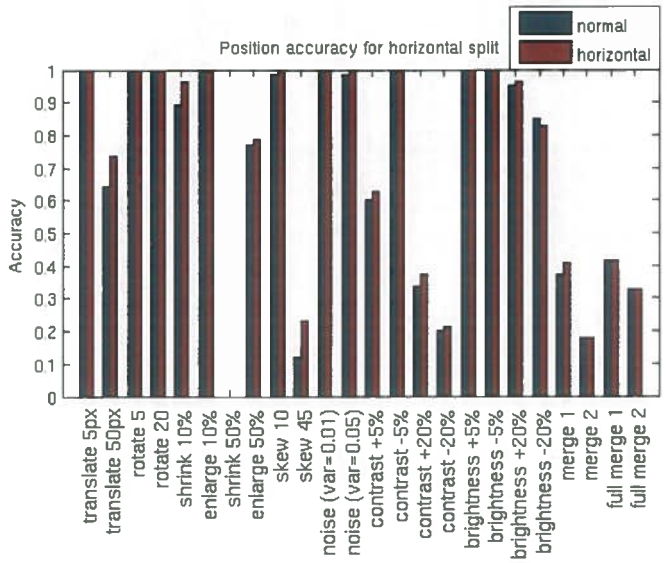


Figure 6.7: Results of testing each deformation using horizontal split

Segmenting each frame into sections was hoped to improve accuracy by capturing the differences within the frame that is present in normal photographic composition (i.e. sky and landscape, the top half and the lower half). There were three variations where the content was split horizontally, vertically and into four quadrants. Figures 6.7, 6.9, 6.11 show the results of these features. The corresponding

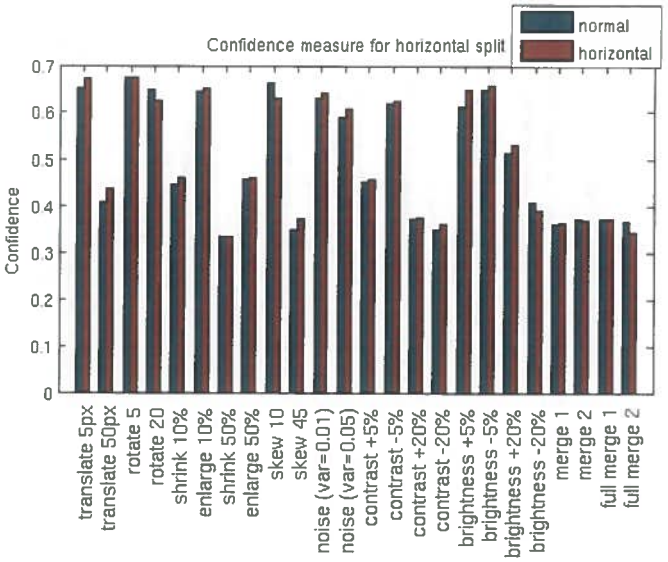


Figure 6.8: Confidence each deformation using horizontal split

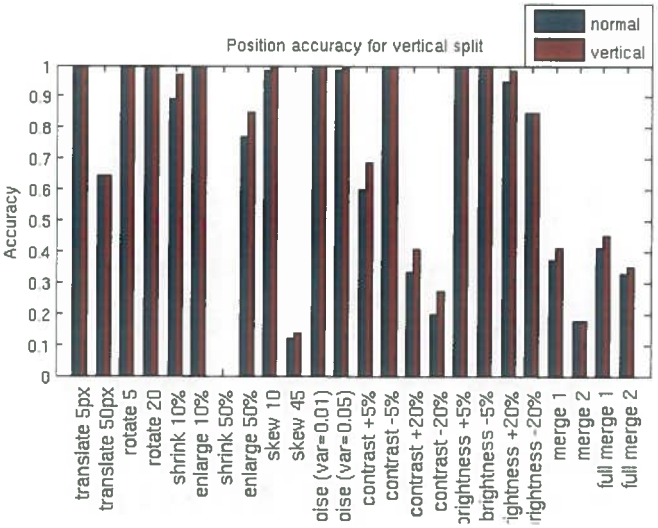


Figure 6.9: Results of testing each deformation using vertical split

confidence is shown in figures 6.8, 6.10 and 6.12.

Doing this created 2 BCSs (4 for quadrants) which were compared in order (top to top, etc.) and the average dissimilarity of these was used for the final dissimilarity. The results of all three variations show improvement on all deformations with the exception of the brightness reduction by 20%. This dropped in the horizontal split and quadrants and it remained the same for the vertical split. The associated confidence only dropped in the horizontal split while it increased for the vertical split and quadrants.

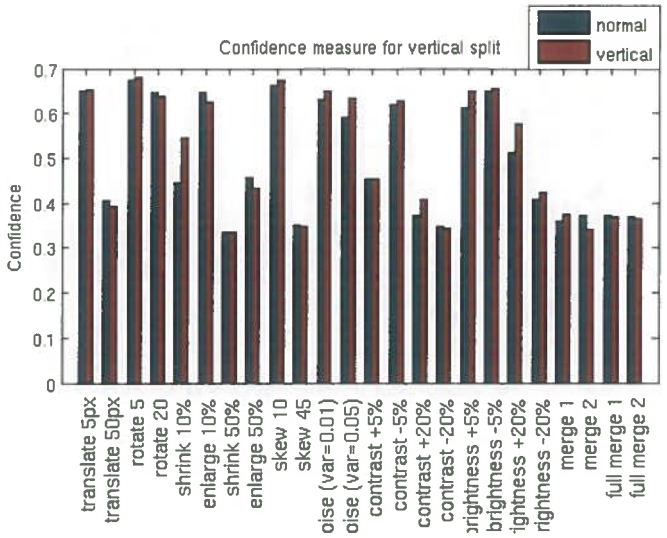


Figure 6.10: Confidence each deformation using vertical split

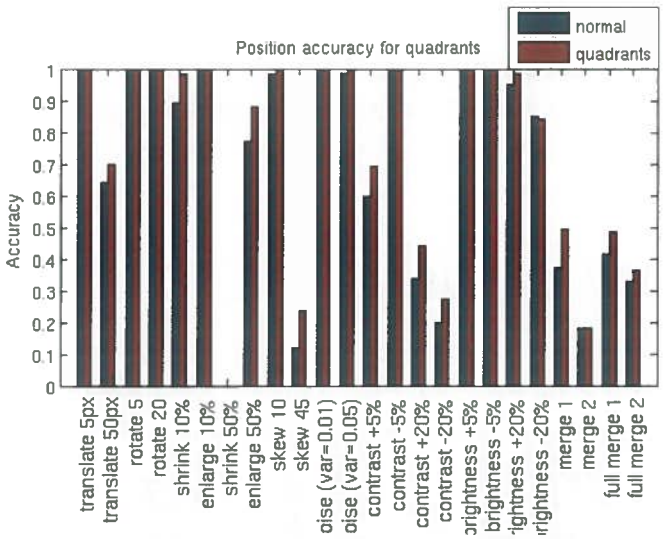


Figure 6.11: Results of testing each deformation using quadrant split

The reason for this could be that because the brightness reduction is one of the least confident results and each split of the image captured the noise created in slightly different ways. If this wasn't the case then there would be a trend, for better or worse, across all the variations. The same effect can be seen in the confidence for enlargement as it only decreases for the vertical split.

There is a confidence drop for the 20° rotation in both horizontal and vertical splits. This is not present in the quadrant split but this can be explained by inspection of the data. When split in two (either horizontally or vertically) noise

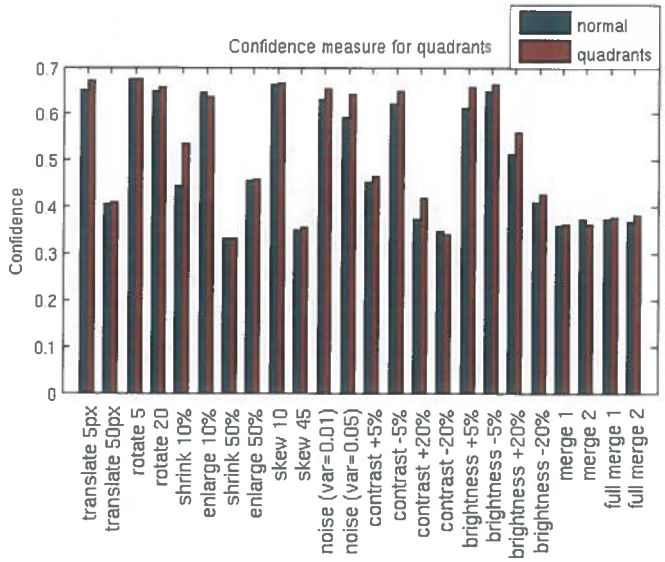


Figure 6.12: Confidence each deformation using quadrant split

is introduced from the black pixels, lowering the confidence of the results. This is not present in the quadrants as two quadrants possess less noise than the others, allowing them to match better with the original video.

6.5 Colour Exclusion

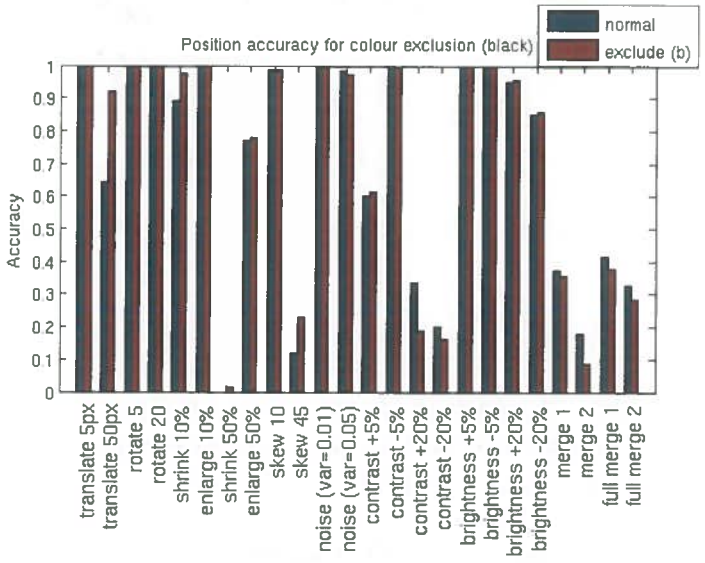


Figure 6.13: Results of testing each deformation excluding black pixels

Excluding black pixels was thought to counteract the noise produced by the translation, rotation, shrink and skew deformations. The noise is created by the black pixels left after these transformations, as they are counted by the histogram like all others in the frame. Figure 6.13 shows the results and figure 6.14 shows the confidence.

Reassuringly, it produces more accurate results for the translation, shrink and skew (rotation was unchanged at the maximum accuracy of 1). The respective confidences (including rotation) also increased. This clearly shows it was effective solution for the desired deformations.

However, it has a negative effect on the major noise and contrast deformations. This is most likely because the deformations introduce black pixels (by noise and increased contrast) which are then excluded from the histogram.

It does not explain why reduced contrast is affected but the cause of this drop might simply be that the reduction in contrast changes black pixels in the original video to a lighter grey. When this occurs, the histogram is slightly altered (with less pixels in the bin where pure black is) which adds more dissimilarity when compared with the original (which would have more pixels in the bin containing pure black pixels) and thus lowering the result. This would also explain the drop in confidence for these deformations.

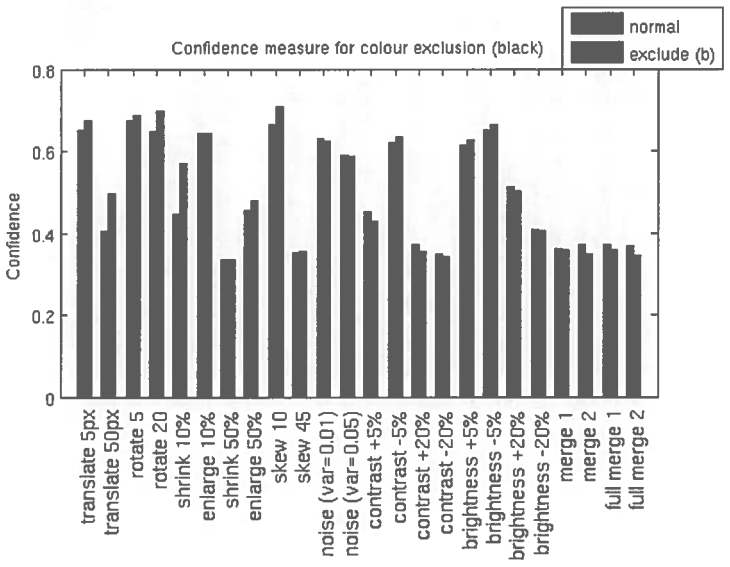


Figure 6.14: Confidence each deformation excluding black pixels

The exclusion of white pixels acted as a counterpart for the exclusion of black pixels. It was hoped this would improve the accuracy of the increased contrast and brightness deformations as pixels are more likely to be shifted up to pure

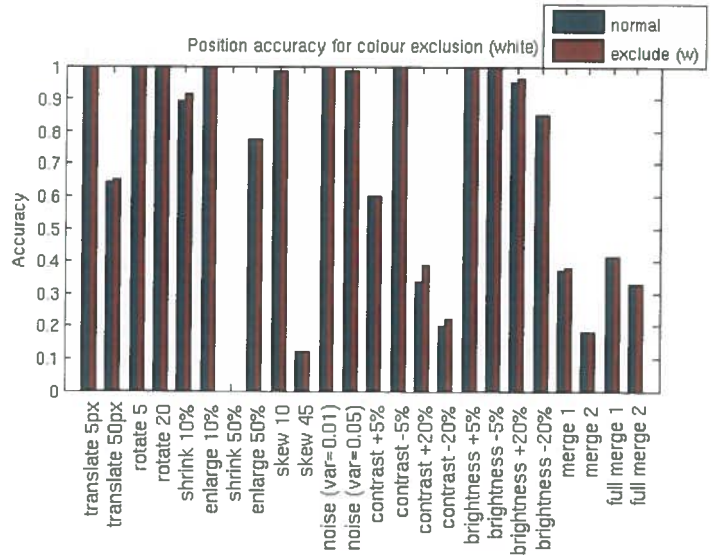


Figure 6.15: Results of testing each deformation excluding white pixels

white. Along with this, if the remaining deformations are unaltered in accuracy, it shows that excluding a specific colour is typically not detrimental to the result.

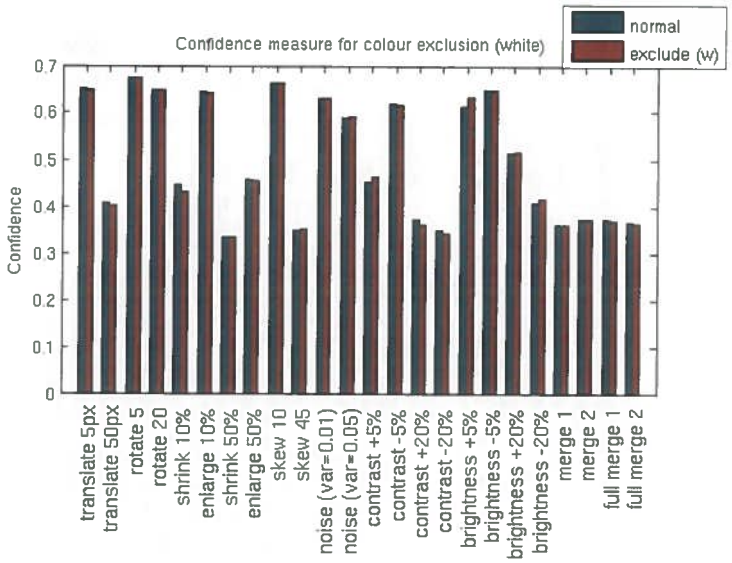


Figure 6.16: Confidence each deformation excluding white pixels

The results of this test (in figures 6.15 and 6.16) show that it slightly improves the accuracy of contrast (both the increase and decrease of 20%) and 20% brightness increase. The confidence for the contrast, however, is lowered so the increase may be again be a product of the noise introduced. Confidence in brightness shows

that the algorithm manages to tolerate this deformation.

This feature also increased the accuracy of translation by 50px and shrinking of 10%. Confidence was lowered for translation and shrinking so the increase may only be because it was affected by the increased similarity brought on by the exclusion.

With this information, it cannot be certain that a colour exclusion always secures the existing accuracy but for the case where it aims to counteract noise (by excluding black) or be more tolerant of increased brightness and contrast (by excluding white), it provides some improvement.

6.6 Weighted Histogram

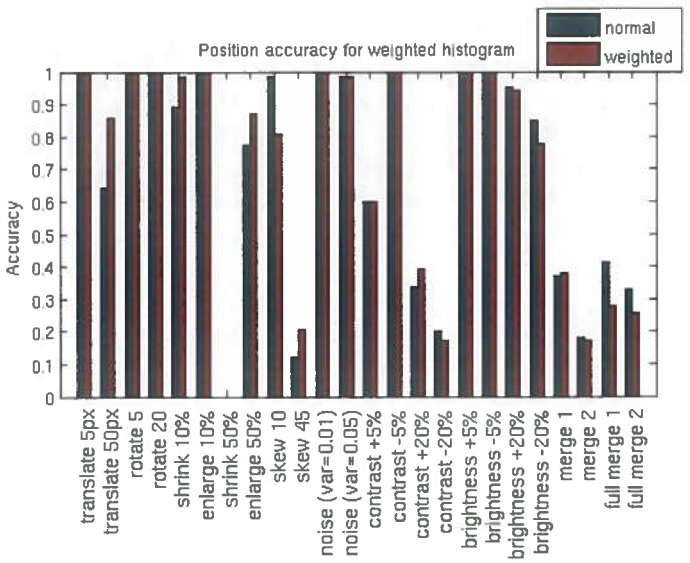


Figure 6.17: Results of testing each deformation using weighted histogram

By applying weights to the centre area of each frame, it was hoped that the weighted area would then have a more prominent presence in the representation, improving results for translation, rotation, scale (both shrinking and enlargement), and skew.

The results (found in fig. 6.17 and 6.18) show that this was the case and for translation of 50px, shrinking by 10%, enlargement by 50% and skew by 10°, the accuracy and confidence was increased (except for skew where it decreased slightly). The reason confidence for both skew deformations decreased is most likely the difference in the size of the weighting area used for the skewed image.

The skew transform changes the dimensions of the image from the standard and as the mask used is determined by the size of the image, this will be different in the skewed videos compared with all other transformations which maintain the size of the original frame.

On average, confidence is slightly reduced for the transformations which may explain the differences in accuracy for the major contrast and brightness deformations.

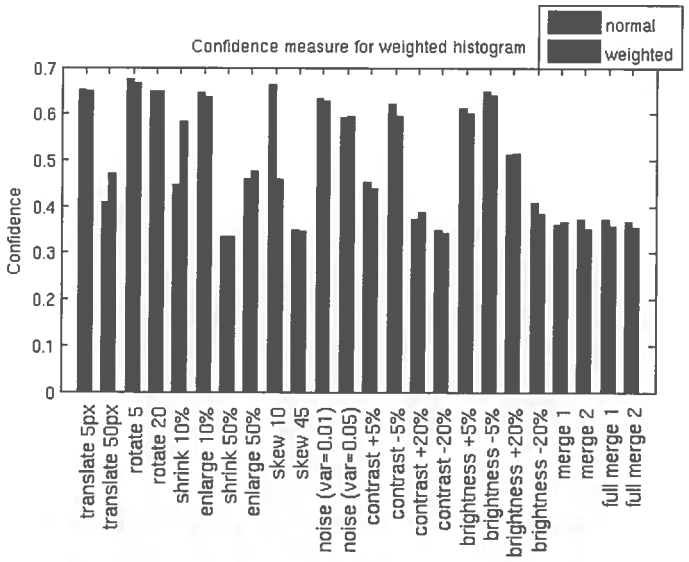


Figure 6.18: Confidence each deformation using weighted histogram

6.7 Summary

Although each experiment only targeted a specific subset of deformations, it is nevertheless interesting to know if the average accuracy and confidence over all deformations improves upon the basic implementation. Overall, there was an increase in accuracy in 6 of the 9 tested features. The following table shows the average accuracy and confidence.

	Av. Position Accuracy (%)	Av. Confidence (%)
Normal histogram	69.23	49.77
Extended histogram	67.92	47.79
HSV colour space	69.43	47.06
View invariant histogram	50.21	40.52
Vertical split	71.58	50.80
Horizontal split	70.98	50.25
Quadrant split	73.07	51.50
Weighted histogram	69.49	49.27
Colour exclusion (black)	69.76	50.90
Colour exclusion (white)	69.73	49.73

These experiments attempted to provide improvement across the range of deformations but there are a few that were never improved. The attempts to increase the accuracy of 50% shrinking and skew of 45° failed and this is because there was too much noise introduced while at the same time, a lot of the data was lost.

The merged videos also showed no real improvement. Some of the investigated features managed to increase the accuracy but this was not that significant and the confidence was still very low throughout. The reason this deformation was so problematic was that the representation was not designed to cope with the mix of data.

Further results produced are given in Appendix E. These show the percentage of videos found in the top n (out of the 10 tested) for each deformation. Each feature tested is plotted on these graphs.

7. Discussion

There was clearly an improvement in the detection of deformed copies using some of the methods investigated. First, there is the original improvement over UQLIPS. As mentioned in the implementation section, the reason behind this appears to be a difference in either the calculation or representation between our implementation and UQLIPS. The exact difference is unknown as there is no way to check the algorithm used by UQLIPS to generate a BCS but it is the most likely explanation for increased accuracy.

To ensure the increase was not simply due to the different data, the algorithm should be tested on the full UQLIPS database. Then, either by executing a regular search (original video as a query) or running the deformation tests again, the new results produced could be analysed to see if they still improve the data. The reason this was not done in this investigation was that there was not enough time or resources to replicate the full database.

Along with improving the basic results to the effect of deformations, 6 of the 9 experimental features gave further improvement. The best of these was the quadrant approach with an overall improvement of 3.84%. While this is quite low, when compared to UQLIPS it improved accuracy by 15.81%. Confidence also improved from both implementations, up by 1.73% from our implementation (reflecting a 5.1% increase from UQLIPS).

Further research for this experiment could be try increasing levels of frame subdivision. If the level of detail [2] method was applied to this scenario, it may improve results. The basic concept of a level of detail system is that the detail increases when the surface is viewed more closely. Applying this to the quadrants would allow for an initial level match to be performed which could be increased by using 8 (or more) sections of the frame if the dissimilarity was low enough (if it was not then it would not be seen as a match). This method could provide an even more robust set of representations for a video. The main problem with this approach would be that there would be much more data to compare per video.

The worst performance was from the viewpoint invariant histogram. This was disappointing as it has the most promise for improvement of skew. The reason behind this was that the weighting function brought all representations to a smaller range within the feature space. As this created a lot more similarity between all videos, the noise reduced the overall results. To see if the viewpoint invariant histogram could be salvaged, it would be worth investigating and altering the weighting function so that it would not reduce the data quite so severely.

To improve upon the results achieved, combinations of the features could be

investigated. As the exclusion of black pixels gave the next best result after splitting the images, it would be interesting to investigate whether combining the two could offer an improvement upon the quadrant features by excluding black pixels from them.

One area that was not investigated was the content of the histograms themselves. By extending them with scale and rotation invariant moments [6], the results could be improved as it provides more data to assess dissimilarity. Ideally this would not add any excess noise but without full investigation, the possibility could not be ruled out.

The data could also be improved by increasing the subset of videos searched and transforming (and testing) more videos. This would increase the reliability of the results gained as the main concern in this project was that the test set was not large enough to provide definitive evidence of accuracy improvement. As previously mentioned, this would also give further proof of the improvement produced by the basic algorithm.

Another way to increase the reliability of the results gained would be to perform searches in reverse. Rather than submitting a deformed copy in attempt to match it against the original, it would be interesting to see if the original video could match all the deformed copies in a database containing the full set of deformed and original videos. This would also allow for the measurement of precision and recall as there would be a set of copies to locate where as previously there was only one copy.

8. Conclusion

The aim of this project was to investigate how well the BCS representation could handle deformed video copies. This involved using the deformed copy as a query to search a database containing only one original copy. It represents quite a tough search problem as there is only one target as opposed to a set of related targets usually assessed in a search problem.

The implementation of the BCS algorithm was successful and this was verified by comparison of the results from UQLIPS for the transformed videos. A check was also performed by searching for all original videos using an original video and our implementation found all originals with 0 dissimilarity. Our implementation also resulted a basic improvement which was theorised to be because of the difference between calculation and/or representation of the data.

After performing tests on the experimental features, the results showed improvement for some deformations but for others the experiment failed to increase accuracy. However, the hypothesis was proven to be true: different features improved overall accuracy of the deformations in 6 of the 9 investigated features. Overall, the best result produced an improvement of accuracy from UQLIPS' 57.26% to 73.07%. This is a vast improvement but compared to the normal histogram on our implementation it represents an improvement of 3.84%.

We have outlined possible avenues for improvement of this result and the next step in this investigation would be to experiment with the feature combinations suggested in the discussion. The database size and number of transformed videos should also be increased to ensure a reliable result.

The data used was the weakest point of this investigation. While the results show improvement, it is only across a small subset of the data. The validity of the results would be much better if the test set was increased and more rigorous testing was performed.

There were also some deformations where there was no improvement or only a slight improvement to an already low accuracy. While the attempts to improve the effect of 50% shrinking and skew of 45° failed, it was thought that the combination of noise and data loss made these deformations particularly troublesome to match.

Merged videos were also not improved greatly but as none of the experiments aimed to fix this problem specifically it was not expected that an improvement would be made. However, if the frame data (histograms) was clustered and each cluster was given a BCS, this could be improved. It would, however, also increase the complexity of matching the clusters as each cluster would have to be compared

to all other clusters. Using a technique like locality-sensitive hashing [5] may aid this process as it uses hashing functions to locate similar data (which could then be matched in the normal way).

Sadly, contrast was was also not improved to a large extent. Only the HSV color space aimed to fix this but it was not successful. Histogram equalisation may improve this as it can counteract the effect of contrast adjustment.

8.1 Future Work

The topic of video copy detection has a large scope for experimentation and we were only testing the ability to detect copies of deformed video data.

Along with the improvements suggested, there is more research that could be done to fully test the BCS representations tolerance to transformed videos. One extension would be to assess the accuracy for videos which have more than one type of deformation applied. For example, increasing contrast and skewing the video would provide a challenge to the system as these deformations are normally less accurate when used in isolation. Making a random selection of deformities would increase this challenge and it would be interesting to see how well BCS can handle them.

One of the drawbacks of the experimentation was that the videos were manually deformed. These don't accurately simulate some of the deformations present in real world data. Testing a system of original videos (e.g. films) with their illegally recorded counterparts would be beneficial to check that BCS can handle the data that it would encounter if put to use in a commercial system.

Regarding the video data itself, investigating the effect of video length would be a worthwhile study as it has not been researched previously for the BCS representation. The techniques mentioned previously to increase the accuracy for merged videos could also be applied here so that the video is segmented into smaller sets of frames.

While there are many areas for improvement, this project managed to achieve increased accuracy in the detection of deformed video copies and by implementing the ideas proposed, these results could be improved much more.

Bibliography

[1] Donald A. Adjeroh, M. C. Lee, and Irwin King. A distance measure for video sequences. pages 25–45, 1999.

[2] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, Department of Compuer Science, University of Utah, 1974.

[3] Chih-Yi Chiu, Cheng-Hung Li, Hsiang-An Wang, Chu-Song Chen, and Lee-Feng Chien. A time warping based approach for video copy detection. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, pages 228–231, 2006.

[4] Justin Domke and Yiannis Aloimonos. Deformation and viewpoint invariant color histograms. In *British Machine Vision Conference*, pages 11–509, 2006.

[5] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *The VLDB Journal*, pages 518–529, 1999.

[6] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Information Theory, IEEE Transactions on*, pages 179–187, 1962.

[7] Yu-Gang Jiang and Chong-Wah Ngo. Visual word proximity and linguistics for semantic video indexing and near-duplicate retrieval. In *Computer Vision and Image Understanding*, pages 405–414, 2008.

[8] I. T. Jolliffe. *Principal Component Analysis.* Springer, 2002.

[9] Changick Kim and B. Vasudev. Spatiotemporal sequence matching for efficient video copy detection. In *Circuits and Systems for Video Technology, IEEE Transactions on*, pages 127–132, 2005.

[10] Richard J. Larsen and Morris L. Marx. *An Introduction to Mathematical Statistics and Its Applications.* Prentice Hall, 2006.

[11] David G. Lowe. Object recognition from local scale-invariant features. pages 1150–1157, 1999.

[12] Heng Tao Shen, Beng Chin Ooi, and Xiaofang Zhou. Towards effective indexing for very large video sequence database. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 730–741, 2005.

[13] Heng Tao Shen, Xiaofang Zhou, Zi Huang, Jie Shao, and Xiangmin Zhou. Uqlips: A real-time near-duplicate video clip detection system. In *Proceed-*

ings of the 33rd international conference on Very large data bases, pages 1374–1377, 2007.

[14] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477 vol.2, 2003.

[15] Alvy Ray Smith. Color gamut transform pairs. In *Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, pages 12–19, 1978.

[16] Gilbert Strang. *Introduction to Linear Algebra.* SIAM, 2003.

[17] Lloyd Nicholas Trefethen and David Bau. *Numerical Linear Algebra.* SIAM, 1997.

[18] Karthikeyan Vaiapury, Pradeep K. Atrey, Mohan S. Kankanhalli, and Kalpathi Ramakrishnan. Non-identical duplicate video detection using the sift method. In *Visual Information Engineering, 2006. VIE 2006. IET International Conference on*, 537-542.

Appendix A. Test Video Overview

Figure A.1 shows the 10 videos that were used for deformations. Each row shows 6 frames, one taken every 10 seconds from the video. These include one video filmed mainly in black and white, two similar videos (both adverts for Woolworths) and a cartoon-style video.

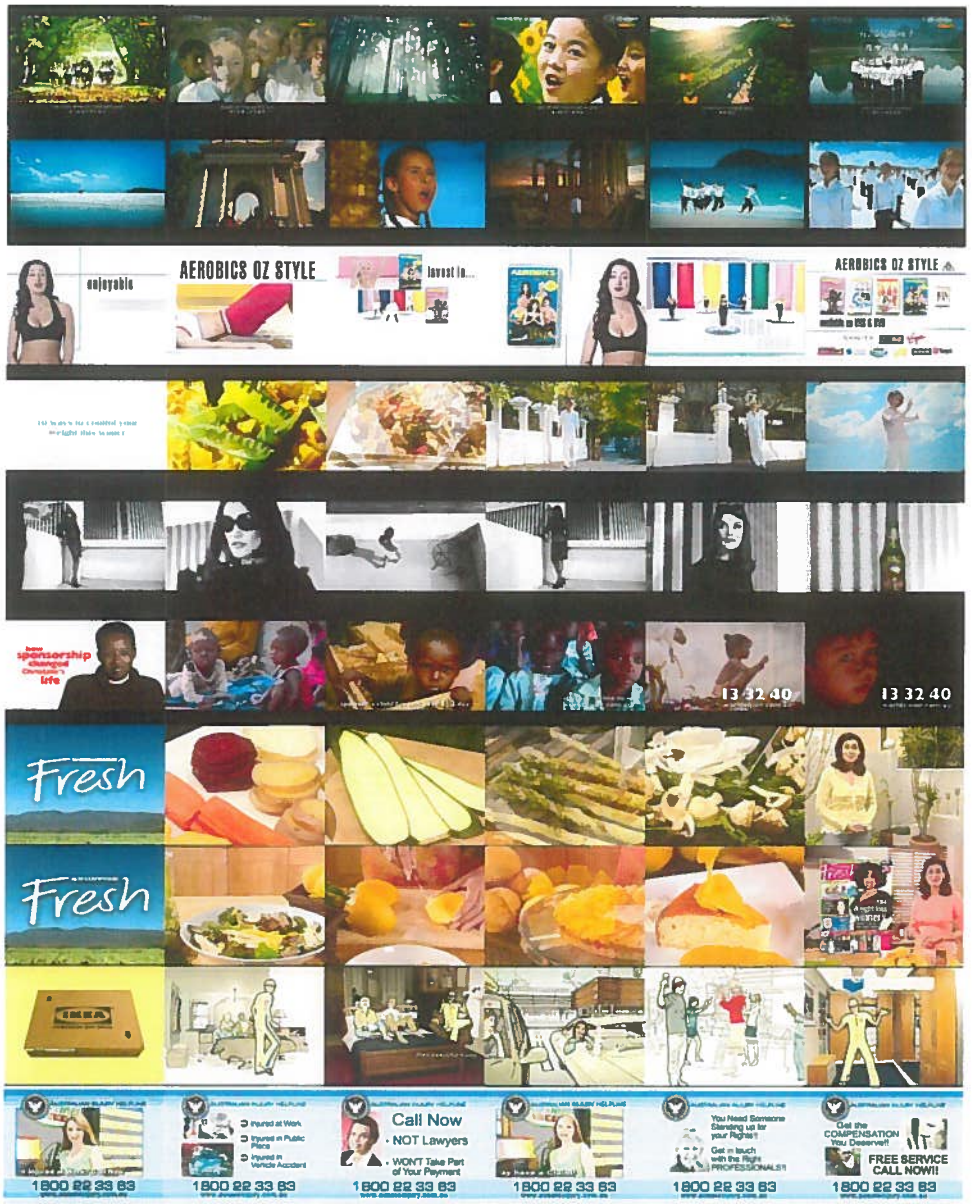


Figure A.1: Test video overview

Appendix B. Histogram Calculation Algorithm

To calculate a histogram for a give frame, the following algorithm was used:

```
// image is an array with RGB values
im = data[w][h][3];
size = 4;
// divide every value (each channel for each pixel)
// by the size of the bin (e.g. 64),
// floor it and add 1 to get range 1 to size
histim = floor(im/floor(256/size)) + 1;

hist = int[size][size][size];
// this loops through each bin and
// fills it up
for i = 1 to size
  for j = 1 to size
    for k = 1 to size
      // count the totals pixels with the
      // current permutation
      hist[i][j][k] =
        count(im[:][:][1] == i &
              im[:][:][2] == j &
              im[:][:][3] == k);
// flatten array into a (size*size*size)x1 vector
return reshape(hist, size*size*size);
```

N.B. The size value is altered to create the extended histogram

Appendix C. Weighted Histogram Calculation Algorithm

To calculate the weighted histogram the following algorithm was used:

```
// image is an array with RGB values
im = data[w][h][3];
// divide every value (each channel for each pixel)
// floor it and add 1 to get range 1 to 4
histim = floor(im/64) + 1;

// setup a binary mask
mask = int[w][h];
radius = min(w,h)/3;
centre = {w/2, h/2};
for i = 1 to w
  for j = 1 to h
    // if the pixel is in a circle with given radius
    if in_circle(w, h, radius, centre)
      // set to histim so count doubles within mask
      mask[i][j] = histim[i][j];

// initialise histogram to 0
hist = int[4][4][4];
// this loops through each bin and fills it up
for i = 1 to 4
  for j = 1 to 4
    for k = 1 to 4
      // count the totals pixels with the current permutation
      hist[i][j][k] =
        count(im[:][:][1] == i &
              im[:][:][2] == j &
              im[:][:][3] == k) +
        count(mask[:][:][1] == i &
              mask[:][:][2] == j &
              mask[:][:][3] == k);
// flatten array into a 64x1 vector
return reshape(hist, 64);
```

Appendix D. RGB to HSV Conversion Calculation

To convert from RGB to HSV, the following formulae are use:

$$r = R/255$$
$$g = G/255$$
$$b = B/255$$
$$max = max(r, g, b)$$
$$min = min(r, g, b)$$

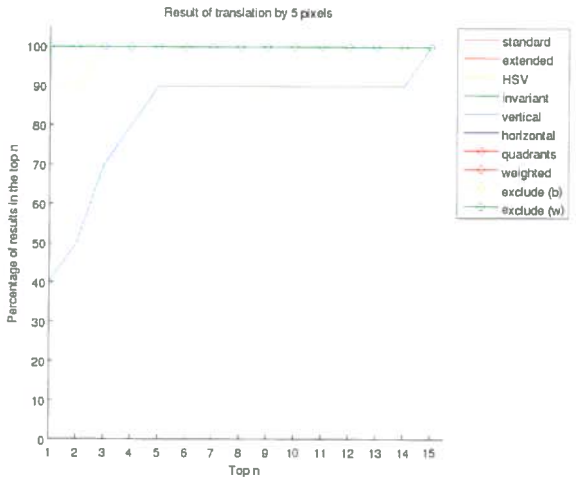
$$h = \begin{cases} 0 & \text{if } \max = \min \\ (60° \times \frac{g-b}{\max - \min} + 360°) \mod 360°, & \text{if } \max = r \\ 60° \times \frac{b-r}{\max - \min} + 120°, & \text{if } \max = g \\ 60° \times \frac{r-g}{\max - \min} + 240°, & \text{if } \max = b \end{cases}$$

$$s = \begin{cases} 0, & \text{if } \max = 0 \\ \frac{\max - \min}{\max} = 1 - \frac{\min}{\max}, & \text{otherwise} \end{cases}$$

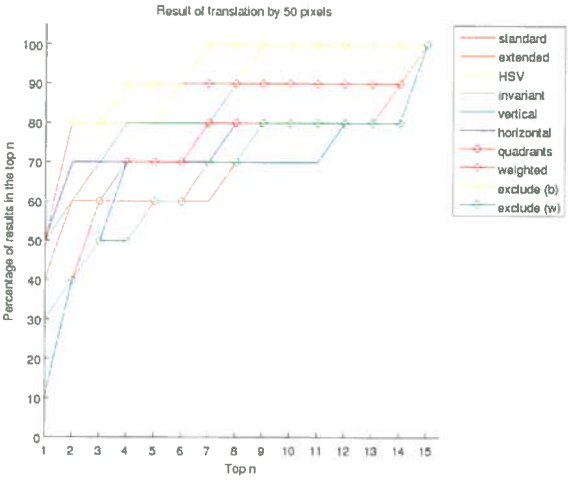
$$v = \max$$

Appendix E. Further results

This section includes the plots of percentage found in the top n (where n varies from 1 to 15) for each feature tested.

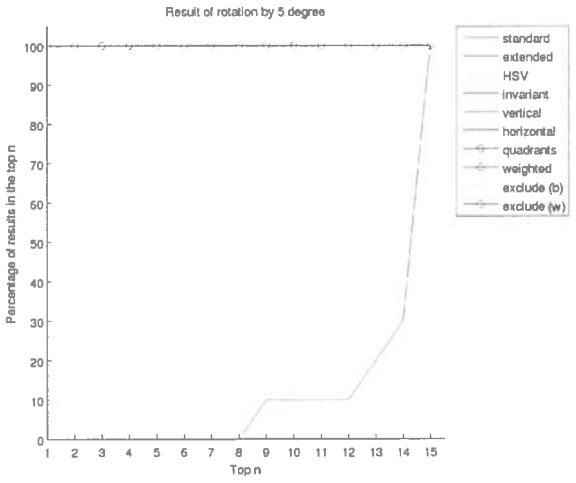


(a)

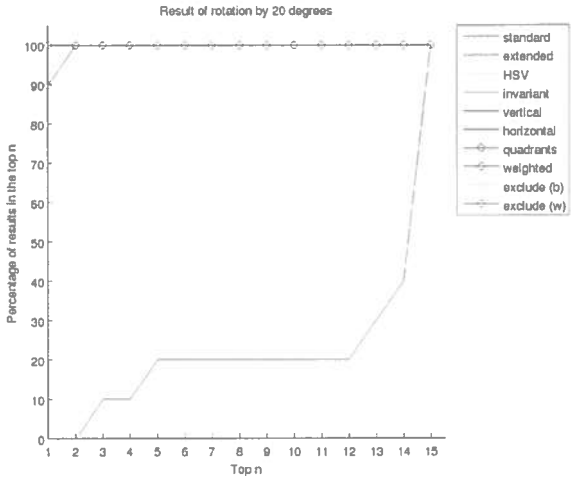


(b)

Figure E.1: Translation of 5px (a) and 50px (b)



(a)



(b)

Figure E.2: Rotation of 5° (a) and 20° (b)

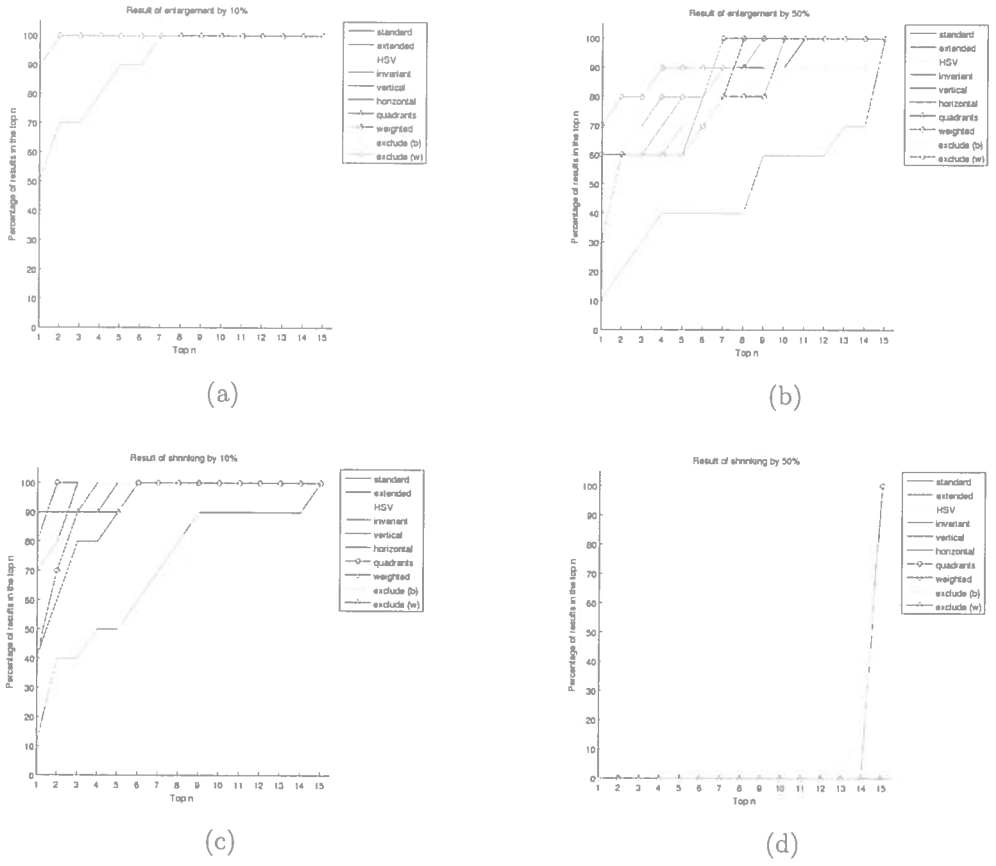
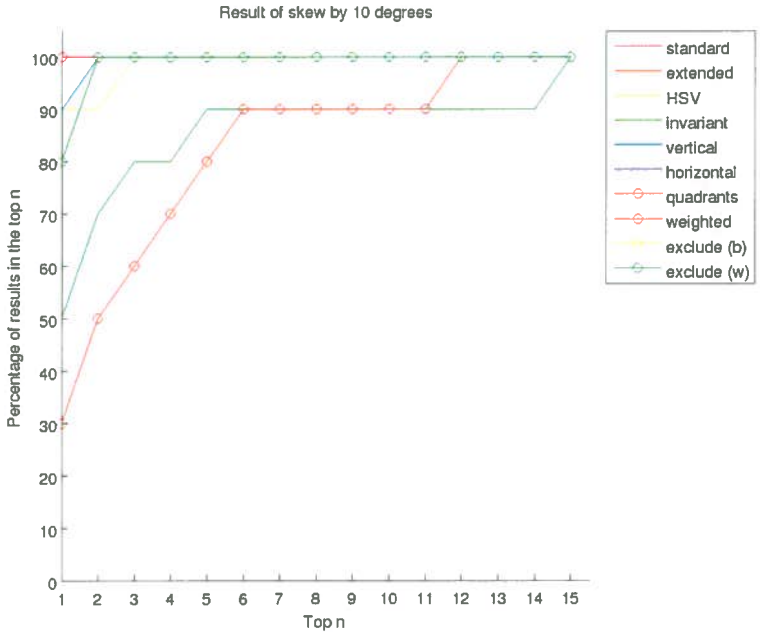
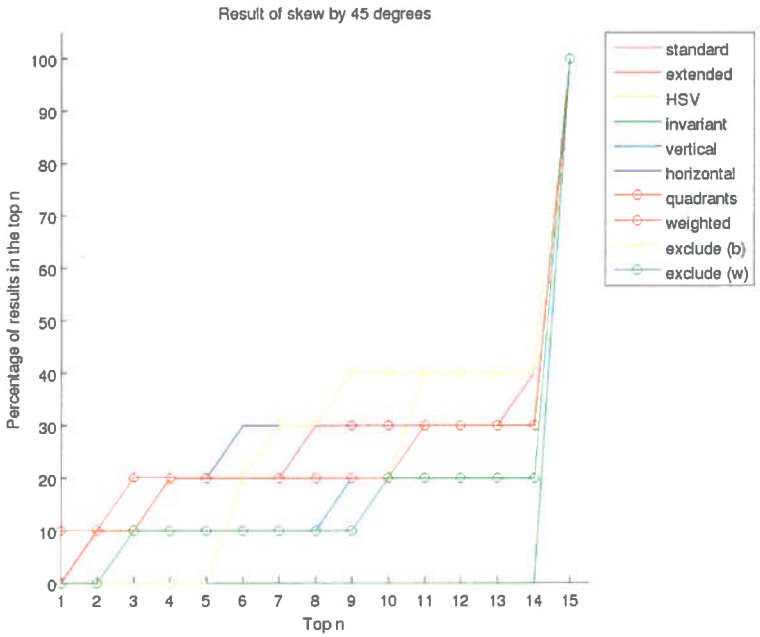


Figure E.3: Scaling of 10% (a), 50% (b), -10% (c) and -50% (d)

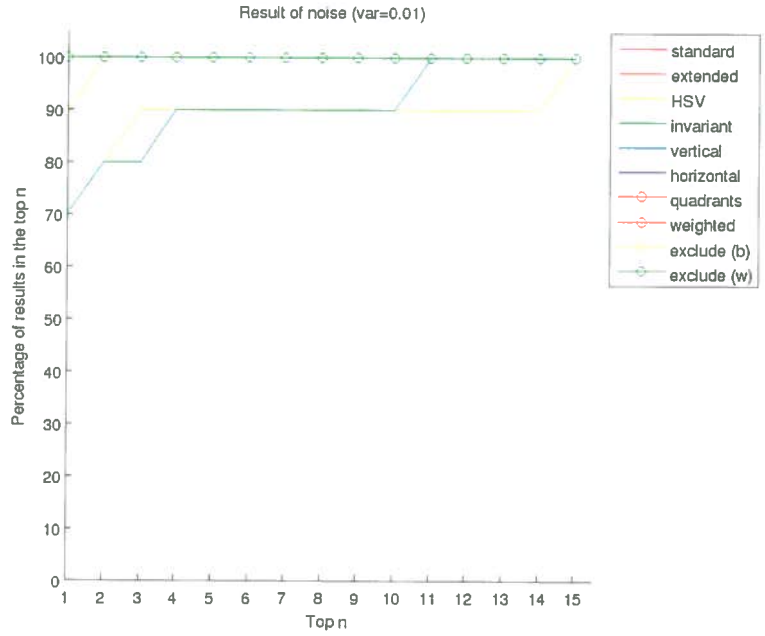


(a)

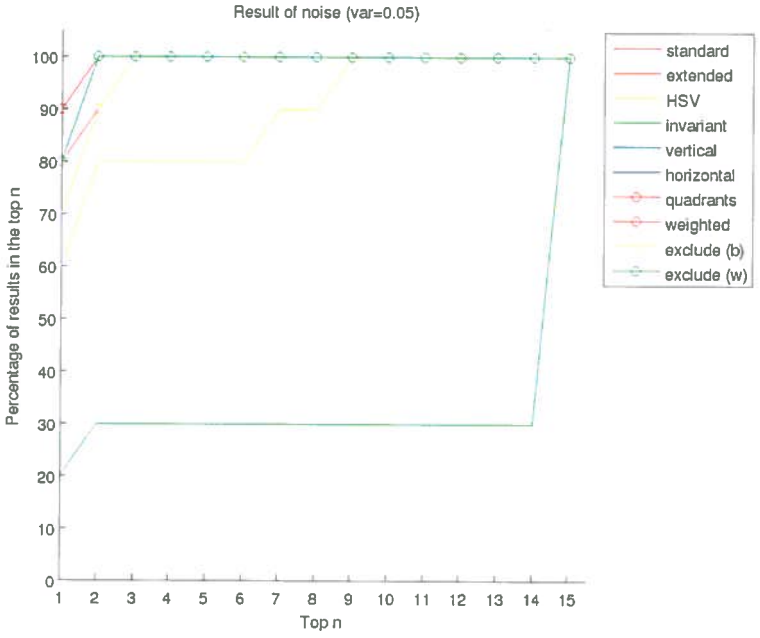


(b)

Figure E.4: Skew of 10° (a) and 45° (b)



(a)



(b)

Figure E.5: Noise with variance 0.01 (a) and 0.05 (b)

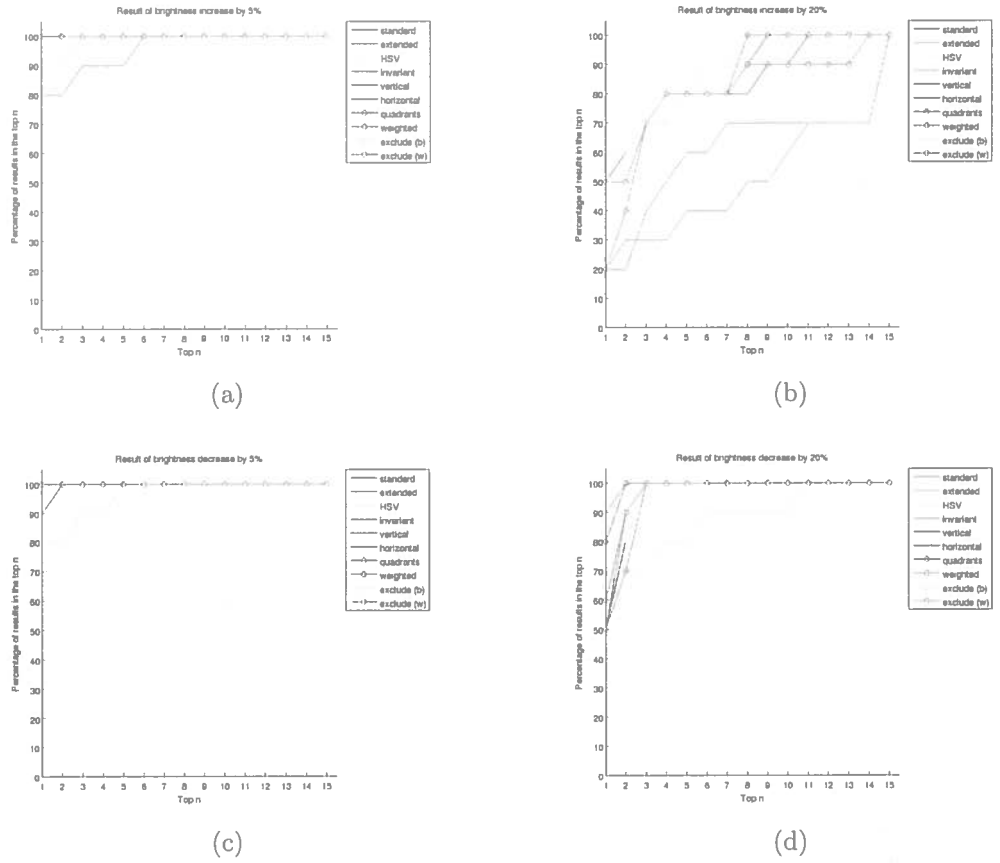


Figure E.6: Brightness decrease of 10% (a), 50% (b) and increase of 10% (c) and 50% (d)

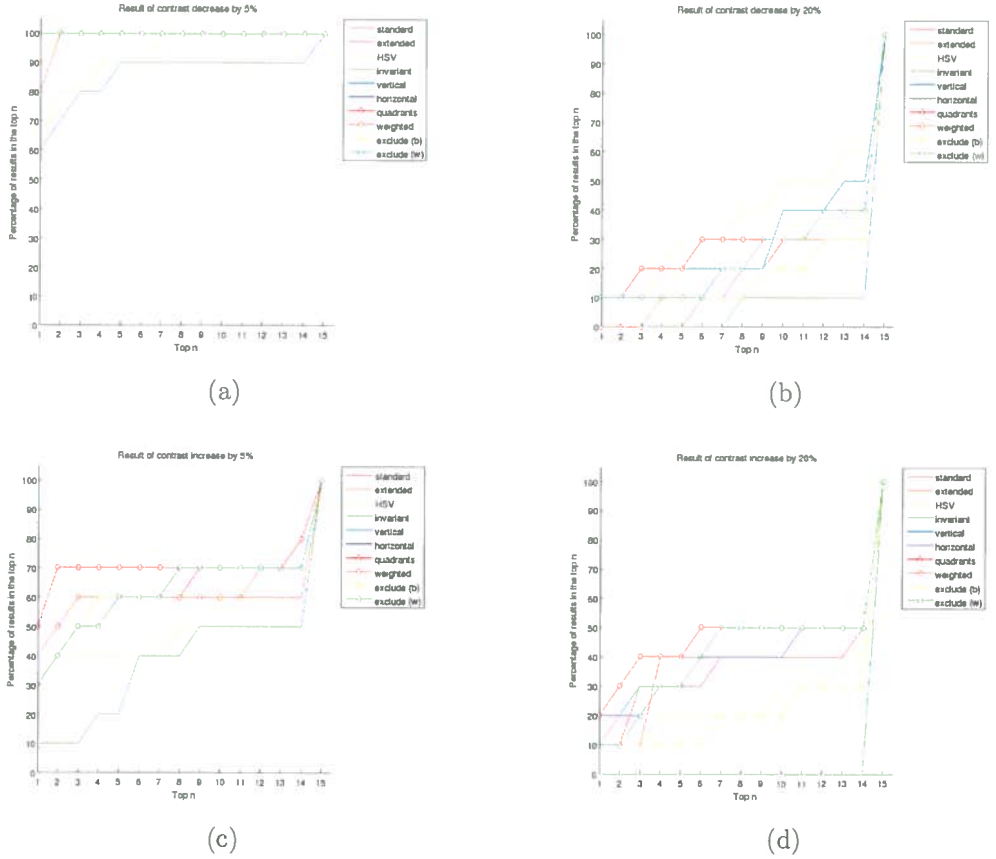
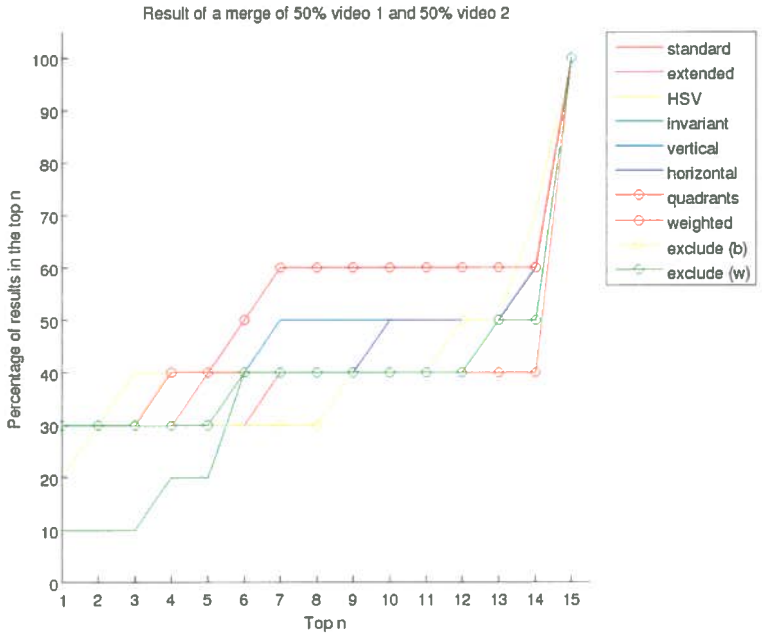
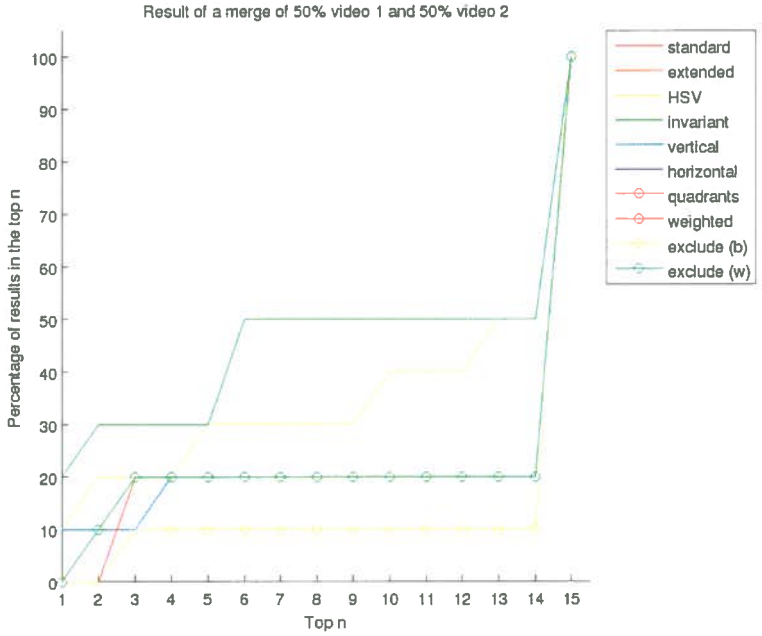


Figure E.7: Contrast decrease of 10% (a), 50% (b) and increase of 10% (c) and 50% (d)

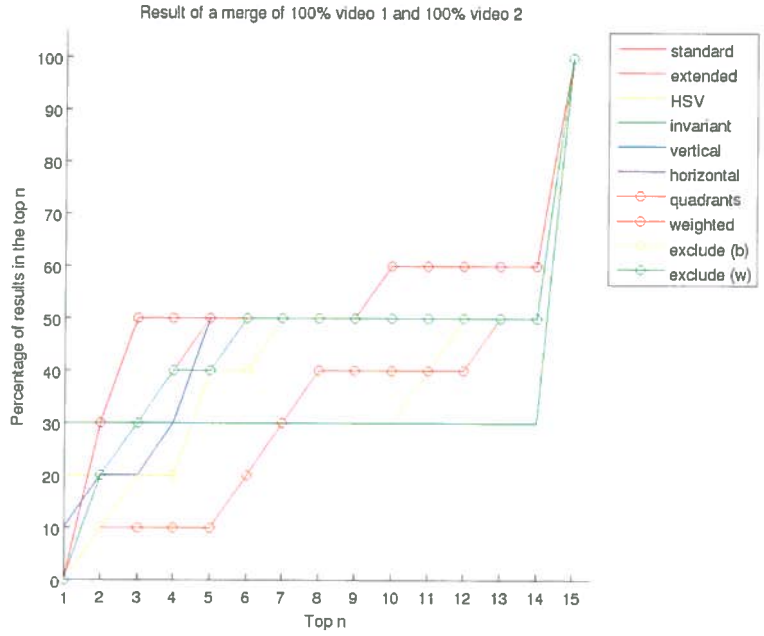


(a)

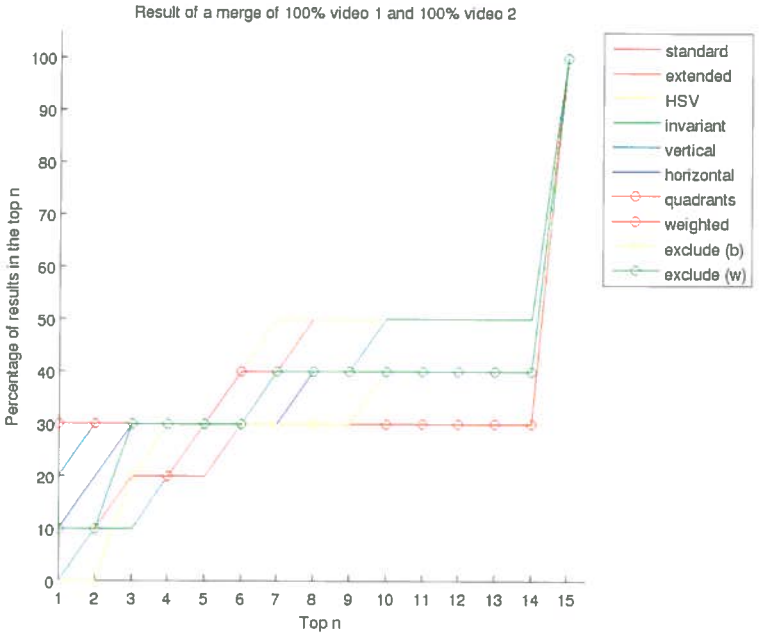


(b)

Figure E.8: 50% merge, targeting the first video (a) and second video (b)



(a)



(b)

Figure E.9: 100% merge, targeting the first video (a) and second video (b)