Content-based Video Retrieval for Pattern Matching Video Clips

Adam Jaamour

Bachelor of Science in Computer Science with Honours
The University of Bath
May 2019

Content-based Video Retrieval for Pattern Matching Video Clips

Submitted by: Adam Jaamour

COPYRIGHT

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed: Adam Jaamour

Abstract

Your abstract should appear here. An abstract is a short paragraph describing the aims of the project, what was achieved and what contributions it has made.

Contents

List of Figures

List of Tables

Acknowledgements

Add any acknowledgements here.

Chapter 1

Introduction

Chapter 2

Literature Survey

2.1 Content-Based Retrieval

Content-based retrieval is a type of visual search technique where large databases of either images or videos are queried to find the closest match to a query image or video. Although this project focuses on content-based video retrieval, also known as CBVR[1], image retrieval techniques, also known as CBIR[2], will be discussed as well due to their relevance in video retrieval.

This section will first review the different video retrieval methods, starting from text-based retrieval to content-based retrieval, before addressing the various challenges that exist in video retrieval, such as the difficulty caused by the temporal aspect of videos compared to images, and the complications of targeting mobile devices for such a system.

2.1.1 Video Retrieval Methods

Drastic advances in video capturing technology have caused important amounts of unstructured data in the form of videos to be produced in recent years. This has lead to a high-demand to develop new efficient solutions for processing this data, with video retrieval being the answer

Throughout the years, video retrieval has improved in parallel with the breakthroughs in video recording devices. Early video retrieval techniques used a text-based approach where the system accepted text input to search the database of videos (**?**), as seen in Figure 2.1 *a*. For example, the user would input the query *"De Niro"*, which would return all movies in which Robert De Niro starred or *"Coppola"* to find all movies directed by Francis Ford Coppola. Unique aspects of the video clip such as movie credits or sports scores

[1] Content-Based Video Retrieval
[2] Content-Based Image Retrieval

were often analysed using OCR[3] technology (**?**). The query text was then compared to a video file's content, such as colours, shapes, texture, luminance or objects, or to the file's metadata[4], such as the video title, author, date, content description, commentaries, captions or keywords (**?**; **?**; **?**). However, these techniques were highly inefficient compared to content-based techniques as they often relied on manually noted annotations and textual descriptions to find similarities for matching the query video to a video in the database and did not make use of the actual visual content that describes a video.

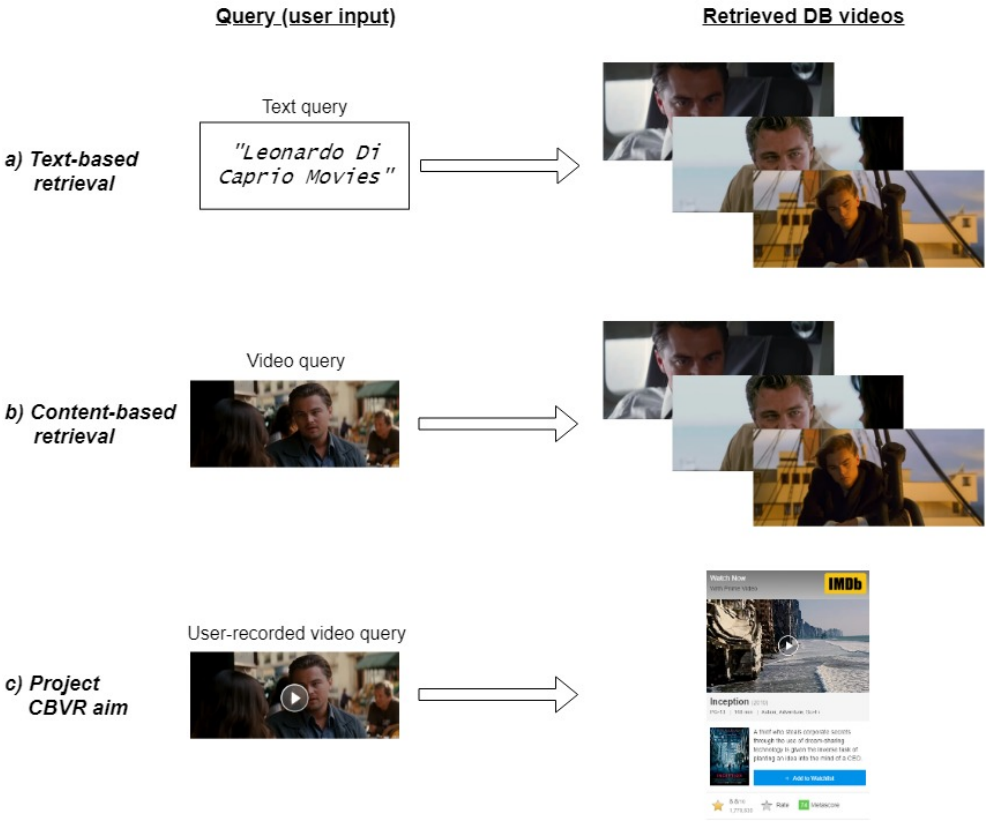


Figure 2.1: Illustrations of a text-based content-retrieval system *(a)*, a content-based video retrieval system *(b)*, and the project's desired CBVR system *(c)*.

Content-based retrieval techniques quickly replaced text-based retrieval techniques towards the end of the 20th century by making use of the visual content to compute similarities between videos (**?**). Instead of accepting a text string, the system takes a video as input to extract its visual contents, as shown in Figure 2.1 *b*. According to Petković et al. (**?**), this visual content

[3]Optical Character Recognition
[4]The data associated to a video file

can be broken down into three different categories:

- *Raw data*, which corresponds to individual raw video frames and the video file's attributes such as the frame rate per seconds, the number of bits per pixel or the colour model used.
- *Low-level visual content* consists of the visual features that describe a video. This content includes colours, shapes, textures and motion. Low-level visual content can be extracted into features, which are defined in Section 2.2.2, using a wide variety of techniques that are used to detect similarities between videos (**?**) and later pattern match them.
- *Semantic content* contains the high-level concepts that are present in a video. These high-level concepts can be described as objects or events using the features. To extract semantic content from a video, a grammar of rules for objects must be provided. An example of an object rule could be "if the shape is round, the colour is orange and the object is moving, then that object is a basketball".

In comparison to raw data, low-level visual content provides more relevant information to extract from a video. Additionally, semantic content extraction adds an unnecessary layer of complexity compared to low-level visual content extraction as it requires domain knowledge and user interaction (**?**). Therefore, this project will focus on using low-level visual content to extract information about the video and compute the similarities between the query video and database videos. It is important to note that this project's goal differs from classic CBVR systems where a list of videos is returned (see in Figure 2.1 *b*), as it must return a specific video that corresponds to the query video (see Figure 2.1 *c*). To improve the pattern matching accuracy phase, raw data (e.g. audio) and metadata (e.g. captions) may be used to improve the pattern matching accuracy (**?**).

2.1.2 Temporal Aspects of Videos

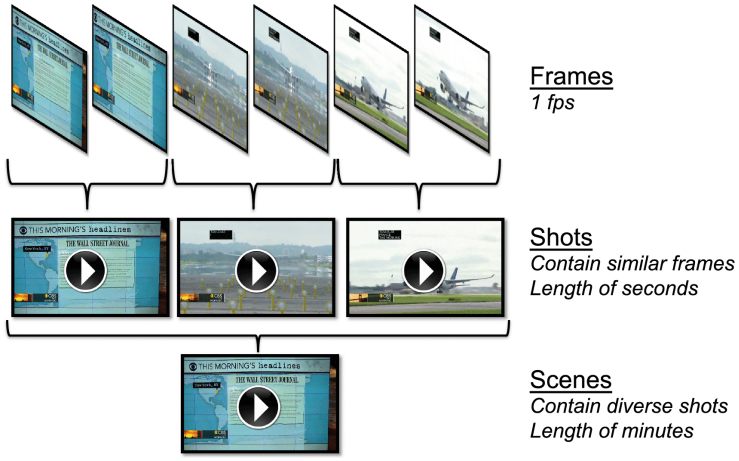
Temporal Structure

The most important difference between content-based image retrieval and video retrieval lies within the temporal aspect of the video. Naturally, the temporal aspect of a video clip stores supplementary information about the content, including dynamic low-level visual content e.g. an object's motion, and semantic content e.g. actions and events. According to A. Araujo et al. (**?**), a video's temporal structure can be subdivided into three units, as shown in Figure **??**:

- *Frames* correspond to the smallest temporal unit of a video file. A single segment of a video is referred to a frame. Frames are also used to describe

the numbers of stills in a second e.g. 24 fps corresponds to a video made up of 24 stills per second.

- *Shots* are grouped sequences of visually similar frames. They are usually described in seconds.
- *Scenes* are a collection of shots which are related based on the action and objects present in the shot, thus giving them a semantic aspect. Scenes lengths are generally calculated in minutes rather than seconds.



Because this project will attempt to provide a CBVR solution targeting databases of feature-length movies, a fourth video temporal structure category relevant to this project can be added to Araujo et al.'s list:

- *Movies* can be described as a large group of scenes that are used to tell a story. Movie durations commonly range from one to three hours.

Challenges

Multiple challenges can arise in CBVR in contrast to image retrieval. The temporal aspect of videos adds a new dimension of complexity when extracting visual information. Indeed, low-level visual content describing images includes colours, shapes and textures, whereas the low-level visual content describing videos can include motion. As mentioned previously, videos are a made up of frames, which make up shots when a combination of similar shots are played in succession. This means that videos hold information about motion, such as the trajectory of objects. However, this poses unique challenges to the algorithms used to extract motion.

Because videos are made up of numerous frames, usually around 24 per second, there are almost no differences between two consecutive frames. Bradski et al. (**?**) clearly points out how the pixels describing an object in one

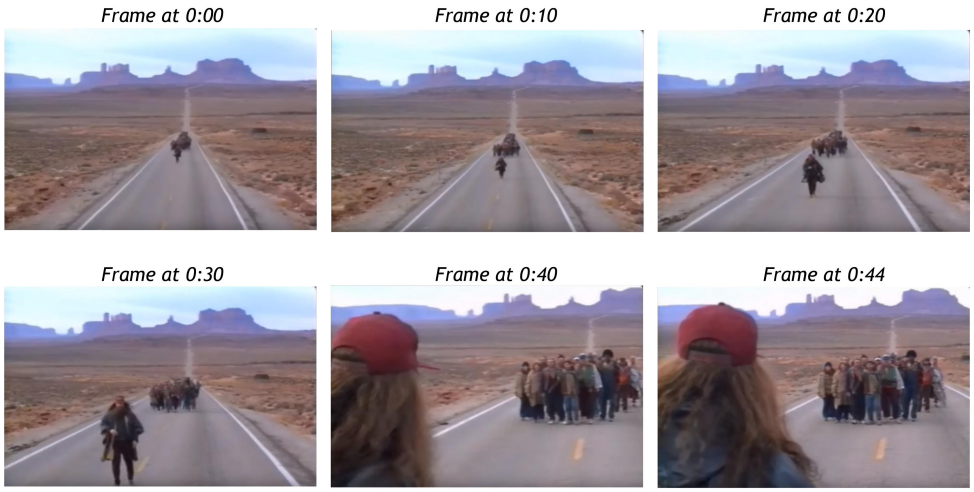


Figure 2.2: Frames from the famous running scene in Forrest Gump extracted at intervals of 10 seconds. Video frames courtesy of *"Forrest Gump long run scene"* YouTube video available online: `https://youtu.be/QgnJ8GpsBG8?t=325`.

frame will remain the same in the next frame, except for the edge pixels perpendicular to the motion trajectory. Figure 2.2 shows six frames from Forrest Gump's famous running shot, which lasts 44 seconds, making up a total of 1056 frames. Each frame in the figure was captured with 10-second intervals, meaning 240 frames separate each. In the first three frames, the group of people running in the background barely moves in 20 seconds (480 frames). The pixels that form the group in the shot remain the same for all of the frames between the 3 samples, with a few new pixels describing the group rather than the road as it advances towards the camera. The same can be said about the red cap in the last two frames. Most of the pixels making up the cap in the fifth frame remain the same in the sixth frame. This example betrays the reason why analysing a video frame by frame would be extremely inefficient when it comes to CBVR. Due to the similarities between consecutive frames, frames should be aggregated (**?**) to describe a shot by using a selection of frames, such as taking the six frames in Figure 2.2 to describe the entire 44 seconds of video.

2.1.3 Mobile Devices CBVR

The aforementioned project aim is for the system to work on mobile devices for two reasons. The first reason, as shown in Figure 2.3 is to allow users to directly use their mobile phone to record the query video by pointing their camera to a screen displaying a movie, which will in turn tell them which

movie is being played. Additional information retrieved from IMDb[5] such as cast, crew, ratings, runtime and synopsis could also be displayed. The second reason is the popularity of mobile devices, which may be due to the improvements made on mobile phones' processing power, allowing more tasks to be carried out through this medium. However, such a system on a mobile device causes many problems regarding the query video recording method and the computational power available on mobile devices.

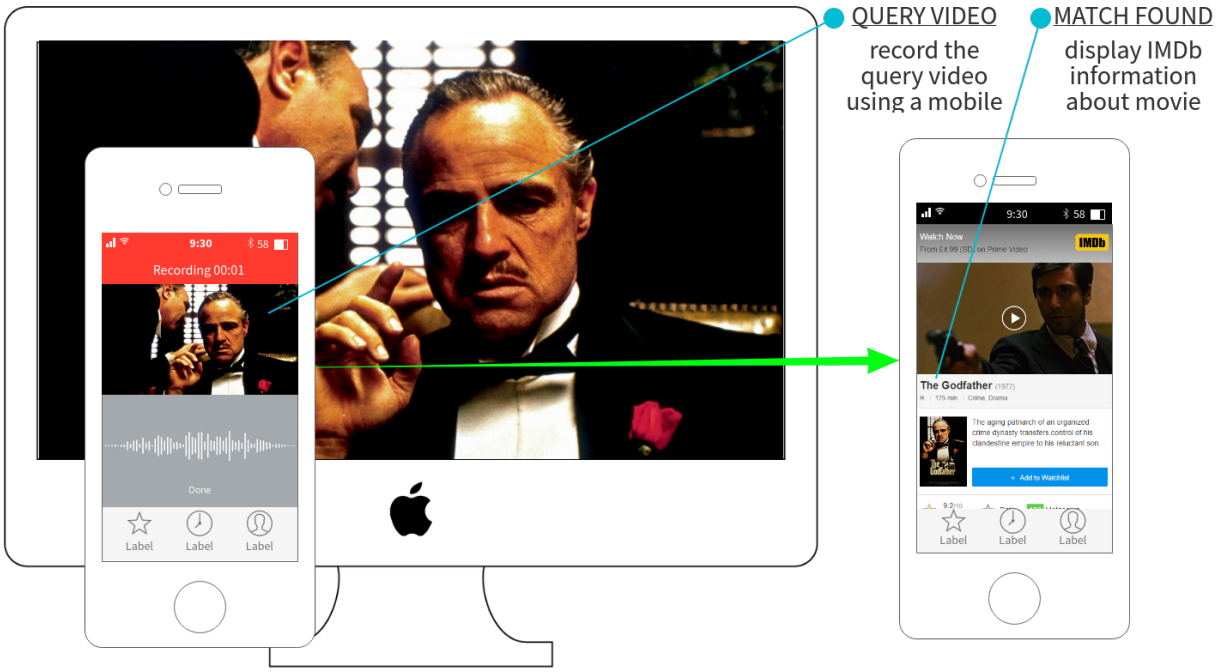


Figure 2.3: Wireframe showing the basic high-level concept of the system.

Query Video Quality

Large visual differences are caused between the query clip and the actual clip stored in the database due to the capture conditions (**?**; **?**) such as:

- Undesired camera movements due to unstable recording e.g. unstable recording, hand shaking.
- Low-quality recording due to poor user recording e.g. scaling and rotation, and due to the environment conditions e.g. lighting, reflections, blurring.

[5]Internet Movie Database

- Video noise because of the camera sensor.
- Decoding artefacts causes by various file compression.

These low-quality conditions add difficulty to the pattern matching phase where the similarities between the query video and database videos have to be computed. Indeed, if the query video is very different to the actual video, then the noisy elements of the video query must filtered out. For example, if the recorded video is shaky, then this shaking motion has to be pruned before analysing the recorded clip's motion. However, processing power must be used from the actual visual content extraction and pattern matching phases to be used for video noise filtering.

User Experience

According to Liu et al. (**?**), the majority of mobile device users expect a polished product with quick video query and instant or progressive results, meaning that the searching algorithms must be efficient. However, one of the downsides of mobile devices is the computation power constraints. Despite the improvements of mobile processors, desktop devices still remain more powerful than mobile ones. A solution that Liu et al. suggest is to retrieve the low-level visual content locally on the mobile device, and send the query to a server where the pattern matching will take place (**?**). This allows heavy computations to be off-loaded from the mobile device. Once a match is found, the result is returned to the user on his mobile device. A downside to this approach is the new constraint on network bandwidth rather than computational power.

2.2 Visual Content Extraction

Good matches when comparing low-level visual content can still produce poor results (60% green and 40% blue example) (**?**)

2.2.1 Static Content

Corresponds to content that can be extracted from a single frame from the video.

2.2.2 Dynamic Content

Corresponds to content that requires the continuity between consecutive frames to be analysed.

Features

Definition

Features correspond to specific visual objects or patterns in an image (or video frame). The low-level visual content aforementioned in Section 2.1.1 can be extracted into features, which store the visual descriptors of the video such as the colours and shapes (**?**) that make an object or a pattern unique. However, only good features features should be extracted from the image. It is therefore important to make the difference between good and poor features to only extract the useful ones. Figure 2.4 shows an image of a 3D-shaped cube, along with coloured patches that make out potential features:

- Blue features
- Green features
- Red features

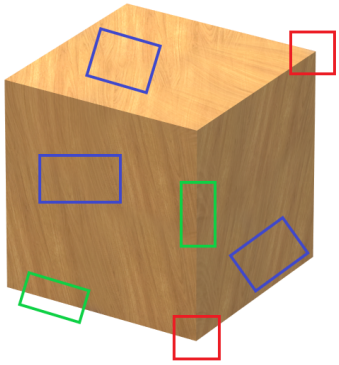


Figure 2.4: A 3D-shaped cube with coloured patches representing potential features.

Once good features like corners are extracted, they are used to compute visual similarities between the query video and the videos in the database. According to Petković et al., two types features can be extracted from a video:

- *Static features* are used to characterise a frame from the video such as colours, shapes or textures. They can be used to describe images and videos.
- *Dynamic features* describe a sequence of frames and therefore store the information regarding the temporal aspect of the video (See Section 2.1.2) such as objects' motions and trajectories.

Multiple techniques exist to retrieve a video's visual information to later compare it to the visual information stored in the database. This section will focus on Fisher Vectors, including the Bag-of-Words model that is necessary to understand Fisher Vectors.

Detection

TODO: Explain corner detection to find good features e.g. Harris Corner Detection

2.2.3 Bag-of-Words Model

TODO:

- Explain how Bag-of-Words model works
- H. Wang's implementation of BoW histogram (**?**)
- State that Fisher Vectors are an improvement on BoW

2.2.4 Fisher Vectors

TODO:

- Fisher Vectors can be used to generate a compact signature for videos, which will then be used for computing similarities between the query video's FV and the database videos' FVs (**?**)

2.2.5 Optical Flow

TODO:

- Optical flow algorithms (**?**):
  - Dense Optical Flow
  - Farneback Polynomial Expression Algorithms
  - Lucas-Kanade Algorithm
- Can calculate optical flow to prune camera movement caused by unstable recording e.g. hand shaking (**?**)

movements.

2.3 Database Videos Pre-Processing

The database of videos can be pre-processed to later optimise the pattern matching phase, enabling the overall system to operate more efficiently.

2.3.1 Key Frames

Y. S. Heo et al. (**?**) suggests only considering key frames in videos to operate on. These key frames would be used for the feature extraction, database preprocessing and pattern matching phases. Key frames are determined based on the difference between two consecutive frames using the histogram chi-square distribution approach (See Equation 2.1).

(**?**) This approach only considers key frames from the query video clip. Key frames are determined based on the difference between two consecutive frames using the histogram chi-square distribution approach (See Equation 2.1).

TODO:

- Explain histogram chi square distribution technique.
- When the amount of differences θ between two frames exceeds a threshold, usually set at the empirical value $\theta = 0.005$, then the current frame is set as a key frame.

$$C(h^{t-1}, h^t) = \sum_{i=0}^{255} \frac{(h_i^{t-1} - h_i^t)^2}{h_i^{t-1} + h_i^t} \tag{2.1}$$

To illustrate the advantage of using key frames, let's use a 3-seconds long shot recorded at 30 fps[6] of a ball rolling on the ground, which would consist of a total of 90 frames. Analysing all the frames individually as stills would be highly inefficient. However, selecting key frames to work on, as depicted in Figure 2.5 where a single frame is chosen for each second, would means that 3 key frames can be used for feature extraction and pattern matching instead of using all of the 90 frames that make up the video.

2.3.2 Thumbnails

In their work, M. Okabe et al. (**?**) generate thumbnails for each video in their database, which are stored as additional data along with the origin video file. The thumbnails for the query video and for the database videos would be generated using the same algorithm in order to create similar results. This technique can be used in parallel to A. Araujo et al.'s (**?**), who states that an initial shortlist of potentially matching videos can be generated before the

[6]Frames Per Second

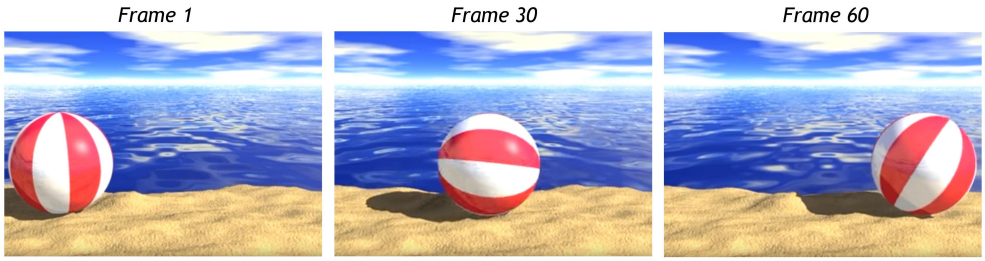


Figure 2.5: Example of frames to sample for low-visual content analysis. The first frame for each second (one frame every thirty seconds) is retrieved for a 30 fps 3-second video of a ball rolling from the left-hand side of the screen to the right-hand side. Video frames courtesy of *"How to Animate a Rolling Ball"* YouTube video available online: `https://youtu.be/cgbLAreElNI?t=130`.

main pattern matching phase. This shortlist can be created by computing the similarities between the query video's thumbnail and the database video's thumbnails. Many advantages can be gained from this small initial step which could be extremely important in the overall system's speed and efficiency:

- Videos that share no similarities to the query video will not be considered at all during the main pattern matching phase e.g. if the query video corresponds to a colourful sunset, then database videos of cloudy environments will be immediately filtered out as it is unlikely that they will match with the query video in the main pattern matching phase.
- This initial step is extremely speedy as it only uses a single still that describes the entire video. Therefore the entire process will not be noticeably slowed down, although this only applies for relatively small databases.
- The database videos' thumbnails will have already been generated during the database's pre-processing phase, which only occurs a single time.

This solution works well for shots, but work less well for scenes and movies (See Figure **??**). Indeed, shots can be characterised by a single image as they are only made up of similar frames. However, the visual content describing the shots that make up scenes (and movies) can be very different from each other. A possible solution could be to create a thumbnail for each shot in a scene, and store each thumbnail in a list. For example, if a scene contains six different shots, then six thumbnails will be generated to describe that scene.

2.3.3 Movie Segmentation

TODO:

- A. Hanjalic et al. (**?**)

2.3.4 Trajectory Analysis

TODO:

- Similar trajectories: if the dynamic features of a video are used to find similarities, then the database of videos can be pre-analysed to detect trajectories (**?**)
- Use of optical flow in this particular case

Chapter 3

Requirements

If you are doing a primarily software development project, this is the chapter in which you review the requirements decisions and critique the requirements process.

Chapter 4

Design

This is the chapter in which you review your design decisions at various levels and critique the design process.

Chapter 5

Implementation and Testing

This is the chapter in which you review the implementation and testing decisions and issues, and critique these processes.

Code can be output inline using `\lstinline|some code|`. For example, this code is inline: `public static int example = 0;` (I have used the character `|` as a delimiter, but any non-reserved character not in the code text can be used.)

Code snippets can be output using the `\begin{lstlisting} ... \end{lstlisting}` environment with the code given in the environment. For example, consider listing **??**, below.

Listing 5.1: Example code

```
public static void main() {

  System.out.println("Hello World");

}
```

Code listings are produced using the package "Listings". This has many useful options, so have a look at the package documentation for further ideas.

Chapter 6

Results

This is the chapter in which you review the outcomes, and critique the outcomes process. You may include user evaluation here too.

Chapter 7

Conclusions

This is the chapter in which you review the major achievements in the light of your original objectives, critique the process, critique your own learning and identify possible future work.

Bibliography

Appendix A

Design Diagrams

Appendix B

User Documentation

Appendix C

Raw results output