Technical Report CS14-11

Approximate Algorithmic Image Matching to Reduce Online Storage Overhead of User Submitted Images

Braden D. Licastro

Submitted to the Faculty of
The Department of Computer Science

Project Director: Dr. Robert Roos
Second Reader: Dr. Gregory Kapfhammer

Allegheny College
2013

I hereby recognize and pledge to fulfill my responsibilities as defined in the Honor Code, and to maintain the integrity of both myself and the college community as a whole.

Braden D. Licastro

BRADEN D. LICASTRO. Approximate Algorithmic Image Matching to Reduce Online Storage Overhead of User Submitted Images.
(Under the direction of Dr. Robert Roos.)

ABSTRACT

Reducing the number of duplicate images uploaded to public servers is an ever more relevant problem as the number of images shared increases dramatically every day. Methods of data reduction such as file expiration dates only lessen this load by a small amount while common methods of image matching are in many cases resource exhaustive, time consuming, or highly inaccurate. This research aims to derive an algorithm capable of identifying near-duplicate images through file hashing, pixel difference, and histogram comparisons. In order to test the feasibility of implementing such an algorithm, a basic photo sharing website has been developed and tested on a fixed collection of images.

Dedication

To Professor Cupper. He was more than just an advisor and professor; he was a member of the Alden family and a father away from home.

Acknowledgements

I would like to thank my thesis advisor, Professor Roos for the time, expertise, and guidance he provided me as I worked through this project. I would also like to thank my family and girlfriend for their support and encouragement that kept me going until the end.

Contents

List of Figures

Chapter 1

Introduction

The purpose of this sample thesis is to show various LaTeX formatting commands. Information about content should be obtained through consultation with the thesis readers and examination of past senior theses. There are no fixed rules governing the number of chapters or their titles—old senior theses and discussions with the thesis readers are the best resources for making such decisions.

Usually, a chapter begins with a paragraph or two that serves as a sort of mini-introduction to the chapter. This is followed by numbered sections created with "`\section{...}`", "`\subsection{...}`", etc.

1.1 Motivation

The first paragraph in each section is not indented—that is a standard style that is enforced by the LaTeX program. It should never be necessary to insert commands to force paragraph indentation.

The thesis should avoid the use of second-person pronouns such as "you" and "your," as well as the use of imperative statements (implied second-person) such as "Look at . . . " or "Consider the following example . . . ". First person ("I," "my," etc.) is sometimes acceptable, but should be used sparingly. (This is an appropriate topic for discussion with the first reader of the project.) Contractions ("don't", "can't", etc.) are considered informal and should be avoided. Common abbreviations such as "math" for "mathematics" or "comp" for "senior comprehensive project" are also informal and not appropriate in the final thesis.

Figures illustrating complex or hard-to-describe concepts are essential. All figures should be fully explained in the text. For example, Figure 1.1 shows the first step in processing a senior thesis. The user files consist of the main file, `AllegThesis.tex`, and zero or more additional files (`ch01.tex`, `ch02.tex` in the figure). Not shown in the figure are image files, the bibliography, and other included components (the ". . . etc. . . . " on the left side of the figure). Typing the command `pdflatex` with the main file name produces a number of additional files, including an `.aux` file (with information such a label references and citations), a table of contents, or `.toc`, file, a printable `.pdf` file, and a number of others, depending on the document.

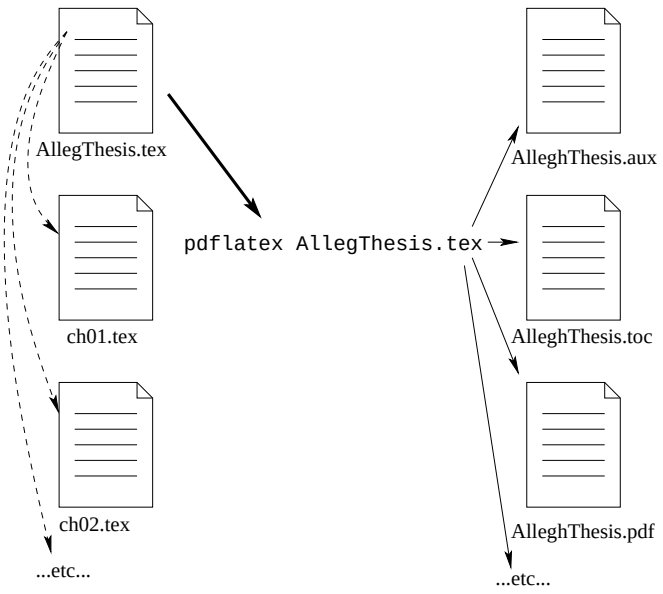

Figure 1.1: The first step in creating a thesis document

1.2 Current State of the Art

There are multiple ways to create *italicized*, **bold-face**, `fixed-width`, and sans-serif fonts, as well as combinations, e.g., ***bold italic*** or *italic sans-serif*. The LaTeX source file for this chapter shows some of the ways to achieve these effects. It is customary to use fixed-width font for program constructs, e.g., "In the Java code accompanying the figure, `n` stands for the number of generations to be simulated by the `evolvePopulation()` method." Variables in mathematical equations are normally rendered in italics, e.g., "In the performance equation, $cpuTime$ is the average time over fifty runs of the program."

It is important to get as quickly as possible to a concise statement of the *thesis*—the main question addressed by the project. This might merit a separate section of the chapter.

1.3 Goals of the Project

This section could also be entitled "Thesis" or something similar.

A formal thesis statement should be a statement about the goal achieved by the project. For a purely scientific project, this is the hypothesis being tested; it should be a *falsifiable* statement, i.e., one that can be disproven through an appropriate experiment. For an applied programming project, it is usually a statement about the feasibility and correctness of the approach used and the advantages it has over other approaches, using suitable metrics. For a survey or study, it is usually a statement regarding the need or usefulness of such a study, its intended audience, and so on.

1.4 Thesis Outline

The introductory chapter usually concludes with a "road map" of the upcoming chapters, e.g., "Chapter 2 reviews a number of past approaches to the problem and summarizes their strengths and weaknesses. Chapter 3 outlines the method of approach used to establish the results."

Chapter 2

Related Work

A typical second chapter deals with a survey of the literature related to the thesis topic. The subsections may be organized in whatever manner seems best suited to the material—chronological, or by topic, or according to some other criteria (e.g., primary versus secondary resources). The examples given in the sections in this chapter are nonsensical in content; they are provided merely to give examples of citing bibliographic references. Resources should be cited by author name(s), not by title. There should be a space between the square brackets of a citation and any preceding words. Thus, "Smith and Jones[17]" is wrong; "Smith and Jones [17]" is correct. If the citation is at the end of a sentence, the period goes after the brackets ("Johnson [23].", not "Johnson. [23]").

2.1 Primary Sources

The earliest work done in widget software is described in the seminal 1986 paper by Smith and Jones [9]. Using a networked array of high-performance computers they demonstrate that widgets with k degrees of freedom can be simulated in time proportional to $k \log^2 k$. At the heart of their demonstration is an algorithm for re-encabulating the widgets using a lookup table that can be updated in real time, assuming that the widgets are non-orthogonal. The question of simulating orthogonal widgets is left open, but the authors conjecture that orthogonality will add at most another factor of k to the performance upper bound.

2.2 Recent Results

A number of papers [2, 3, 10] deal with issues that are peripheral to the orthogonal case, but Dioşan and Oltean were the first to tackle it directly. In their 2009 paper [4], Dioşan and Oltean apply evolutionary techniques to the orthogonal widget case, obtaining empirical results that suggest an efficient algorithm might be at hand. Their approach is characterized by the use of a genetic algorithm to evolve other, more problem-specific evolutionary algorithms.

Chapter 3

Method of Approach

This chapter demonstrates how to include short code segments, how to include pseudocode, and a few other LaTeX features.

3.1 Test Environment

Algorithm 3.1 (from [5]) shows a high-level description of an algorithm. There are many options for the display of pseudocode; this uses the `algorithm2e` package [5], but there are a number of others available at the Comprehensive TeX Archive Network (`ctan.org`). Using any of these other packages might require the additon of one or more "`\usepackage{...}`" commands in the main `AllegThesis.tex` file.

Data: this text
Result: how to write algorithm with LaTeX2e
initialization;
while *not at end of this document* **do**
  read current;
  if *understand* **then**
    go to next section;
    current section becomes this one;
  else
    go back to the beginning of current section;
  end
end

Figure 3.1: How to write algorithms (from [5])

3.2 Experiments

Figure 3.2 shows a Java program. There are many, many options for providing program listings; only a few of the basic ones are shown in the figure. Some thought

must be given to making code suitable for display in a paper. In particular long lines, tabbed indents, and several other practices should be avoided. Figure 3.2 makes use of the `listings` style file [8].

```
public class SampleProg
{
   private int dummyVar; // an instance variable

   public static void main(String[] args)
   {
      System.out.println("hello world.");
      // Avoid long lines in your program; split them up:
      dummyVar = Integer.parseInt("1234")
           + 17 * ("abcde".substring(1,3).length()
           + 11 - 1000;
      // Use small indents; don't use the tab key:
      for (int i = 0; i < 10; i++)
      {
         for (int j = 0; j < 10; j++)
         {
            if (i > j)
            {
               System.out.println(i);
            }
         }
      }
   }
}
```

Figure 3.2: `SampleProg`: A very simple program

3.3 Threats to Validity

Chapter 4

Implementation

Another possible chapter title: Experimental Results

Chapter 5

Discussion and Future Work

This chapter usually contains the following items, although not necessarily in this order or sectioned this way in particular.

5.1 Summary of Results

A discussion of the significance of the results and a review of claims and contributions.

5.2 Future Work

5.3 Conclusion

Appendix A

Java Code

All program code should be fully commented. Authorship of all parts of the code should be clearly specified.

```
/**
 * SampleProg -- a class demonstrating something
 * having to do with this senior thesis.
 *
 * @author   A. Student
 * @version  3 (10 December 2013)
 *
 * Some portions of code adapted from Alan Turing's Tetris program;
 * relevant portions are commented.
 *
 * Revision history:
 *     10 December 2013 -- added ultra-widget functionality
 *     18 November 2013 -- added code to import image files
 */
public class SampleProg
{
   private int dummyVar; // an instance variable

   public static void main(String[] args)
   {
      //================================================
      // This section of code adapted from Alan Turing's
      // Tetris code. Used with permission.
      // http://turinggames.com
      //================================================
      System.out.println("hello world.");

      dummyVar = Integer.parseInt("1234")
           + 17 * ("abcde".substring(1,3).length()
           + 11 - 1000;

      for (int i = 0; i < 10; i++)
      {
         for (int j = 0; j < 10; j++)
         {
            if (i > j)
```

```
            {
               System.out.println(i);
            }
         }
      }
   }

   /**
    * foo -- returns the square root of x, iterated n times
    *
    * @param    x   the value to be interatively processed
    * @param    n   number of times to iterate square root
    * @return   sqrt(sqrt(...())) [n times]
    */
   public double foo(double x, int n)
   {
       for (int i = 0; i < n; i++)
           x = Math.sqrt(x);
       return x;
   }
}
```

Bibliography

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] H. Blum. A transformation for extracting new descriptors of shape. In W. Wathen-Dunn, editor, *Models for the Perception of Speech and Visual Forms*, pages 362–380. MIT Press, Amsterdam, 1967.

[3] James N. Damon. Properties of ridges and cores for two-dimensional images. Technical report, Department of Mathematics, University of North Carolina at Chapel Hill, 1995.

[4] Laura Dioşan and Mihai Oltean. Evolutionary design of evolutionary algorithms. *Genetic Programming and Evolvable Machines*, 10:263–306, 2009.

[5] Christophe Fiorio. algorithm2e.sty — package for algorithms, release 5.0, 2013. `http://www.ctan.org/pkg/algorithm2e`. Accessed 9 Sept. 2013.

[6] Jacob Furst. *Height Ridges of Oriented Medialness*. Ph. D. Thesis, University of North Carolina, Dept. of Computer Science, 1999.

[7] David F. Griffiths and Desmond J. Higham. *Learning LaTeX*. SIAM, Philadephia, 1997.

[8] Carsten Heinz, Brooks Moses, and Jobst Hoffmann. The listings package, 2013. `http://www.ctan.org/tex-archive/macros/latex/contrib/listings/`. Accessed 10 Oct. 2013.

[9] John Smith and Jane Jones. A made-up title of an article in a conference. In Joan Johnson, Sally Simpson, and Dan Douglas, editors, *Proceedings of the Third International Conference on Conferences*, pages 500–509, Paris, 1986.

[10] Justin Zobel. *Writing for Computer Science*. Springer-Verlag, 1997.