UCLA M.S. Comprehensive Exam
Smart Image Conversion

Nicolas Langley 904433991

December 4, 2015

Abstract

Images are one of the most commonly shared forms of data on the internet and are represented by a large number of different file formats with unique characteristics. Users often want images in a specific format based on factors such as size, usage and compatibility. These formats each present the image data differently, yet are often semantically equivalent to the user. This project worked to develop a web browser extension that converts images found on the web to a desired format in-place while browsing. The result is an intelligent and easy way of allowing an user to interact with images from the web in their desired format.

1 Introduction

Images are one of the most widely shared forms of data on the internet. There are nearly 1.8 billion images uploaded and shared every day. Images are consumed for tasks that vary from viewing of family photos to manipulating complex diagrams. As a result, there are a wide range of file formats that are used to store image data and which have specific characteristics that lend themselves to certain tasks.

Many of the available image formats are semantically equivalent in that they represent the same data and there is a a means of translating between the different formats. This notion of equivalence is integral in the value proposition of this project. While many image formats are functionally equivalent, there is still a clear difference between formats and users may prefer one over another. This preference often stems

from usage factors such as desired compression, operating system or color palette support. Users are often tasked with either viewing files on the web in an undesired format or converting downloaded files to their desired format manually.

This project showcases a browser extension that converts images in-place within the browser. The goal is to allow a user control over the format of the image data they view on the web and make content consumption easier.

This report is divided into 4 main sections.

1. Common image formats will be examined and compared to provide some context about why one might be desirable over another.
2. The implementation of the browser extension itself will be detailed. This will include an overview of the development of Chrome Extensions as well as the specific techniques used in the image conversion process. Any complications or other issues will also be detailed in this section.
3. Overview of result of project and use of developed extension.
4. Examination of some related work and potential for future work.

2 Format Comparisons and Trade-offs

2.1 GIF Overview

GIF (Graphics Interchange Format) is a bitmap image format originally developed in 1987 that supports lossless compression. Up to 8-bits per pixel are supported allowing for an image to reference a 256 color palette chosen from the 24-bit RGB color space. The format uses the LZW (Lempel-Ziv-Welch) compression method for data compression. It is possible to store multiple images in a single file and produce simple animations using a control sequence.

2.2 JPEG Overview

JPEG is a compression method based on the discrete cosine transform. The image is transformed from the spatial to the frequency domain where the image data is stored in terms of quantized coefficients. The compression process is generally lossy

and due to the nature of the compression transparency is not saved in JPEG images. It is possible to achieve a large reduction in file size using JPEG that may not be achievable using other lossless compression methods

There are two main file formats that are used to exchange images compressed with JPEG. They are the JPEG File Interchange Format (JFIF) and the Exchangeable Image File Format (Exif). JFIF files are more widely used for sharing across the internet.

JPEG is a desirable format when attempting to reduce the size of a natural image due to its parallels to the human visual system. It is ideal for images that transform naturally into the frequency domain. Large files and photographs are commonly shared in JPEG-based formats

2.3 PNG Overview

PNG is a bitmap image format that supports lossless compression. The format supports full-color and palette-based RGB and RGBA images as well as grayscale images. Data compression is performed in two stages: pre-compression filtering and the DEFLATE algorithm.

The DEFLATE algorithm uses a combination of Huffman Coding and the LZ77 Algorithm to perform bit reduction and eliminate duplicate series of bytes.

PNG as a format is often juxtaposed against JPEG based formats. PNG provides lossless compression and is well-suited for images with sharp transitions and images with large patches of a single color that may not cleanly transform into the frequency domain.

2.4 TIFF Overview

TIFF (Tagged Image File Format) is a bitmap file format that supports both lossless and lossy compression. The format is widely supported by image-manipulation applications. TIFF is a very extensible format that is composed of the Baseline TIFF specifications as well as a number of optional extensions.

Baseline TIFF files must support no compression, CCIT and PackBits compression

which are all forms of lossless compression. TIFF files can be extended to support LZW and JPEG compression which enables the use lossy compression with the format.

TIFF is a very versatile format that supports a wide range of options. It is difficult to support in a general purpose web browser but supports a very diverse set of color palettes and options.

2.5 BMP Overview

The Device Independent Bitmap File Format (BMP) is a raster file format used for storing bitmap images. It is widely used within the Microsoft Windows ecosystem and is most commonly used without compression.

3 Implementation

3.1 Working with Chrome and Chrome Extensions

The choice of tools for implementing this project was the Chrome Extension framework within the Google Chrome browser. Using this toolset and creating an application for Chrome was chosen for a few important reasons and presents some clear benefits as well as some drawbacks.

Chrome Extensions are small applications that run within the Chrome browser. These applications have little to no user interface and are designed to run alongside web pages.

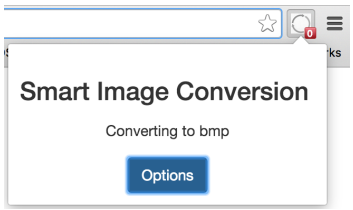


Figure 1: Extension in Chrome browser menubar

Extensions can be written using web technologies such as HTML, CSS and Javascript and are bundled into a single user-installed file. Using a Chrome Extension allows for simple and intuitive modification of the web page DOM (Document Object Model)

through the use of Content Scripts that are injected into the main web page process.

Content scripts are JavaScript files that run within the context of a host web page. They have access to the DOM of that web page and are able to make modifications to the DOM as well as read details about the visited page.

Chrome Extensions consist of a manifest.json file and a collection of HTML and Javascript files. The Javascript and HTML files determine the appearance of certain interface aspects as well as any functionality provided by the extension. For this project, the extension consists of a content-script.js file that handles image conversion within the web page as well as options and popup HTML and Javascript files that allow for a user to manipulate certain conversion parameters.

Chrome Extensions present many benefits in terms of ease of installation, access to a mature set of tools and a clear way of working within web pages. The main trade-off of these extensions is the lack of support outside of the Chrome browser. For this project, the access to tools clearly offset the lack of portability between browsers.

3.2 Output Format Prediction

The extension developed in this project not only performs conversion to a format specified by the user but also allows for specification of certain parameters that are used to intelligently determine an output format. An number of factors that reflect the difference between the different image formats are summarized as options to the user. Based on these selections, an appropriate format is determined.

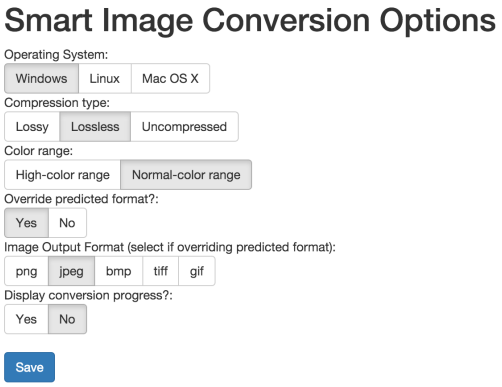


Figure 2: Extension Options Page

The parameters specified by the user are placed in persistent storage within the extension. This is done using the Chrome Storage API and allows for preferences to be saved between sessions and even synced between multiple machines. The extension also presents the user with the option of overriding the generated predictions to specify their own format if desired.

3.3 Converting Images in-place

The main workload undertaken by this project's browser extension is in-place conversion of images within a web page. This work is performed by a content script so that the DOM of the target web page can be modified. This process can be broken into a few main steps:

1. **Identify eligible images for conversion on web page**
   Images within a web page are identified by querying for all <img> tags within the target page. The source location is identified for each image and the file name and initial format for each image is obtained. Image formats are required to fit into the accepted subset of image formats outlined in the format section.

2. **Perform conversion using ImageMagick library and Web Worker**
   Conversion of images if performed the ImageMagick library. ImageMagick is a free and open-sourced software suite for manipulating and converting images. The functionality of ImageMagick is accessed through a set of command-line tools written in C with a wide-array of language specific bindings. The convert module of ImageMagick is used to accurately and efficiently perform conversion

between different image formats.

In order to use this library within a browser client, these tools had to be ported to Javascript using the emscripten tool. [5] Emscripten is an open-source LLVM to Javascript compiler that allows for the conversion of C++ code to the asm.js subset of Javascript. The ported Javascript implementation of ImageMagick is run within the browser using a Web Worker that allows for the code to be executed in the background asynchronously from the main page. The Web Worker object used has a unique file system that is used to store the conversion input, output as well as certain required ImageMagick configuration files. This results in the image conversion being performed without affecting the core usability of the affected web page.

3. **Replace image in-place**
   Once a converted image is obtained, the original image is replaced in-place within the web page. This is done by manipulating the DOM of the target web page within the content script. After the converted image file is retrieved from the convert Web Worker's file system, the Base 64 image URL is extracted. To place the new image within the web page, a new image element is created within the host web page and added to the DOM. Finally, the original input image is removed from the DOM by the content script.

3.4 Converting Images in Parallel

Converting images using ImageMagick in the browser can be a time consuming process depending on the size, initial format and output format of the images involved. In order to speed up the process, features of the Web Worker are taken advantage of to perform image conversion in parallel.

When an image is converted, a Web Worker is created that executes an ImageMagick convert process on the image asynchronously. The Web Worker creates a separate background thread to execute the work and runs a callback function when the work is finished. This paradigm lends itself very nicely to a parallel implementation of image conversion. The resulting performance games are very large when processing multiple images within a web page.

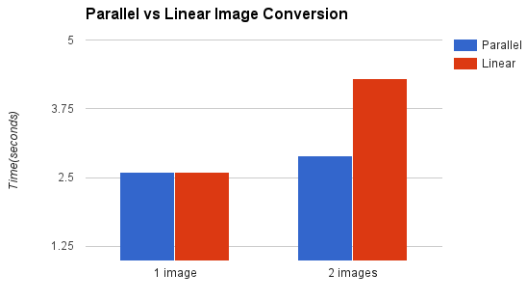


Figure 3: Parallel Conversion Time Comparison

3.5 Complications and Open Issues

In the implementation of this project, there were a number of complications and other issues that arose. Many of the problems encountered are a result of the complex nature and variation within content found on the internet.

This section will outline a number of open and closed issues encountered during the implementation.

1. **Conversion of already compressed images**
   Many of the images found on the web have been compressed and are distributed in a JPEG-based image format. This compression technique is lossy and so some of the original image data will be impossible to retrieve. While there are some obvious enhancements with regards to compatibility between different image formats, conversion to a format that supports lossless compression is bound by the lowest common denominator. This issue is a result of the uploaded formats and is difficult to negate when only processing the images already uploaded.

2. **Handling of misnamed files**
   There is no common naming convention for image files on the internet and this can lead to issues and confusion when dealing with them. The JPEG file format is particularly difficult as there are two vary commonly accepted file extensions .jpg and .jpeg. Issues occurred when using the ImageMagick library for conversion as it only accepts the .jpeg extension. Being unable to process files with a .jpg extension is a large shortcoming that could be addressed by actively renaming files as they are processed. This is unfortunately not handled in the current state of the extension and .jpg files are simply ignored.

3. **Running on complex web pages**
   Web pages are very complex and are composed of both dynamic and static elements in the form of HTML, Javascript and a host of other technologies. The traditional means of displaying images within an web page is to embed a <img> tag within the HTML. The extension developed in this project makes the assumption that all images will be defined in the web page DOM statically. Any dynamic images that are displayed or added to the page are ignored. This is a shortcoming that is difficult to address and is not handled in the current iteration of the extension.

4 Results

The developed extension was tested on a selection of websites that host files of varying formats. Most of the testing was done with images in the PNG and JPEG formats. The performance of converting images varies by the types involved in the conversion as well as the number of images being converted.

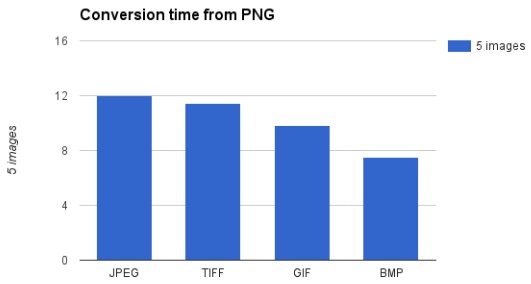
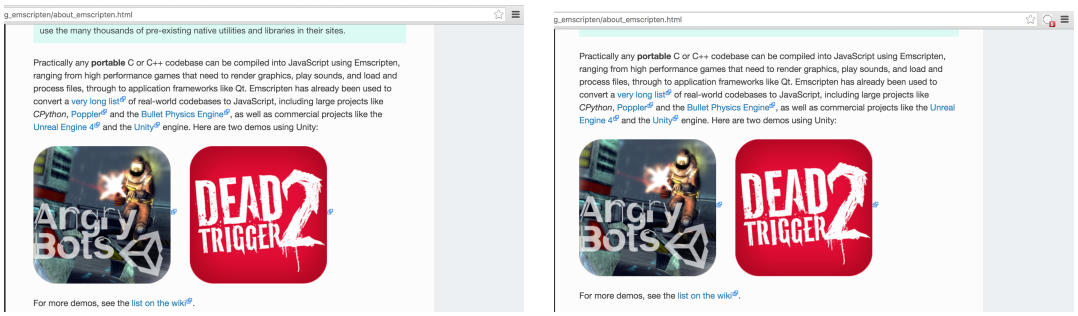


Figure 4: Conversion Time Comparison

Converting images within a web-page is a seamless and occurs in the background. There are some instances where the DOM modifications apply distort the original appearance of the images in the web page but this is dependent on the original page layout.



(a) Before conversion (b) After conversion

Figure 5: Comparison of web-page before/after conversion

The results achieved show that converting images within a web page is time consuming yet yields the desired results. By performing this work in the background it is possible to negate much of the impact seen by the end user while still delivering results.

5 Conclusion

The goal of this project was to develop a way to simplify interaction with images on the web by taking advantage of semantic equivalence between image formats and examining the outcome desired by a user. The resulting Chrome browser extension performs image conversion in-place within a web page. Being able to leverage web technologies to perform all of the conversion allows for a user to define a specified file format and interact with all applicable images on the web in their ideal format. Additionally, the Chrome extension platform removes the need for the user to compile and install any cumbersome conversion libraries and bundles all of the desired functionality into a single file.

There are some clear benefits that can be gained from performing conversion in-place in the web. The developed extension makes the conversion process seamless and can lead to a use reduce in time required for conversion as well as immediate benefits when interacting with a file.

The project succeeds not only in supplying the necessary tools for someone to choose their own format but also a means of using factors related to each format to perform

intelligent format conversion without the user having knowledge of the different types.

6 Related Work

Working with image formats is a commonly explored area. Similar functionality to that described in this project has been proposed in a few different forms and in many different fields of study.

The problem of converting images between formats by considering potential applications usages has been explored extensively within medical imaging where there are a large number of proprietary formats between applications. The challenges involved in converting between commonly used formats for medical data has been explored [1]. This has led to the development of the Medcon toolkit [2] that facilitates easy conversion between medical image formats.

General solutions to this problem have also been presented in the form of a server-based object-database that allows for automatic conversion of a single source file into another specific file format [3]. An implementation of a tool as described above would provide a similar result as that provided by this project, but requires access to the original source file and removes some control from the end user.

This project addresses a unique problem of allowing for automatic conversion of images found within any web page without requiring knowledge of the original source. Many of the other results present a much more complicated and challenging installation and setup process and can not be automated as easily.

7 Future Improvements

There are many improvements that could be made to the browser extension implemented in this project. A selection of potential improvements will be highlighted in this section

1. **Support for a wider range of image formats**
   There are a few formats that are used for sharing images on the internet that the developed browser extension does not support. Two prominent examples of

other formats are JPEG-2000 and WebP. JPEG-2000 is a wavelet-based format that was developed to supersede JPEG for use on the web. WebP is an image format set to compete with JPEG that is a derivative of the VP8 video format. It is supported within Chrome and presents the potential for a large reduction in file size over common formats like JPEG and PNG. Adding support for these formats would increase the usability of the extension.

2. **Generation of easy download links for converted images**
   One of the target use cases for this project is the ability to download images in a users desired format without having to perform any conversion after the file is obtained. While in it's current form, the extension allows for easy conversion, downloading images from a webpage must be done manually by saving each image individually. Keeping a log of the converted images and providing an easy interface for downloading the images to a client machine would be a great addition to this extension.

3. **More in-depth format prediction**
   Format prediction relies on many factors regarding the intended usage of an image. The extension developed in this project predicts a desired output format based on a series of basic parameters. Developing a more in-depth model for predicting the best format for a usage case is a feature that would enhance the usability of the extension. Being able to determine information about the browser itself or what the intended applications a file would be used in after being downloaded would greatly enhance the capability for prediction.

References

[1] Stanchev, P. L. "Medical Image Conversion." National Computer Science Conf., Sofia, Bulgaria. 1994.

[2] Nolfe, E. XMedCon, et al. "An open-source medical image conversion toolkit." Eur J Nucl Med 30.Suppl 2 (2003): S246.

[3] Guck, Randal Lee. "Automatic format conversion system and publishing methodology for multi-user network." U.S. Patent No. 5,911,776. 15 Jun. 1999.

[4] ImageMagick Studio, L. L. C. "ImageMagick." (2008).

[5] Zakai, Alon. "Emscripten: an LLVM-to-JavaScript compiler." Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion. ACM, 2011.