



*Todo list

 **Todo Text:**

Abstract i

 **Todo Text:**

Acknowledgements i



MSc in Computer Science 2017-18

Project Dissertation

Project Dissertation title: Reflectance Transformation Imaging

Term and year of submission: Trinity Term 2018

Candidate Name: Johannes Bernhard Goslar

Title of Degree the dissertation is being submitted under: MSc in Computer Science

Abstract

Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetur. Vestibulum gravida. Morbi mattis libero sed est.

Todo Text:

Abstract

Acknowledgements

Todo Text:

Acknowledgements

Contents

1	Introduction	1
2	Related Work	2
2.1	RTI Theory and Workflows	2
2.2	Fileformats	3
2.3	RTI Viewers	4
A	BTF File Format	6
A.1	File Structure	6
A.2	Manifest	7
A.3	Textures	7
A.4	Options	8

List of Figures

1	RTI Workflow	2
2	RTI Dome	2
3	RTI Overview	3

List of Tables

1 Introduction

Reflectance Transformation Imaging (RTI) was invented by Tom Malzbender and Dan Gelb, research scientists at Hewlett-Packard Labs. It was originally termed Polynomial Texture Mapping (PTM). It is a method of computational photography with great potential for classical archaeology. RTI images (RTIs) are created from multiple digital photographs of an object. A fixed camera position is used in conjunction with a movable light source (or multiple immovable) which hits the object from different positions. Different shadows and highlights result from each light position.

An RTI processing software takes these images and calculates the object's surface per pixel, essentially creating a 2D photograph with embedded reflectance information.

This file can then be viewed with the help of an RTI viewing software, allowing the object to be studied remotely via the user's computer, instead of needing physical access to the object. The software is also able to reveal enhanced or previously unobservable details, e.g. colour, shape, markings or depth of carved lines, which the naked eye could not pick up.

Ancient historians studying material objects benefit from RTI because this software shows how objects were created and subsequently changed. For example captured RTIs of the wooden remains of Roman wax tablets can reveal how they have been inscribed and reinscribed. For statues, particularly Roman imperial portraits RTI can provide evidence for deliberate re-carving of a condemned emperor into his successor.

For video examples and diagrams, the Cultural Heritage Imaging Institute provides a great overview[4].

Other applications of RTI are video games, where RTI can provide self-shadowing for objects, which normal maps could not, though more modern techniques largely replaced RTI in this context.

They are becoming increasingly relevant in a physical based rendering (PBR) context, where RTIs can provide better physical information for objects.

This dissertation is primarily focusing on the application of RTI for ancient historians, but makes provisions for relevance in a PBR context in the future.

oxrti in this document is referring to this project's implementation of an RTI viewer.



Figure 1: RTI workflow, courtesy of CHI[5].

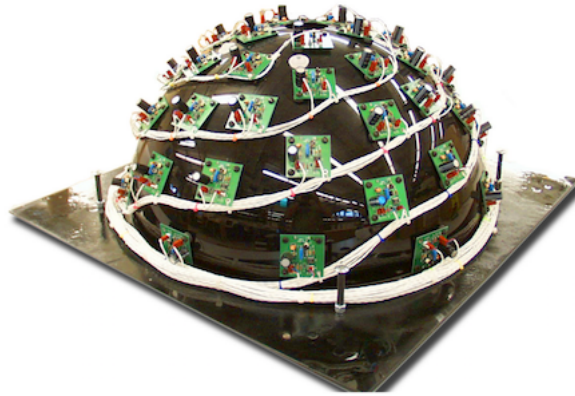


Figure 2: RTI dome, from Malzbender et al.[9].

2 Related Work

This section explores the related work inside the RTI area and explains the underlying principles. The project is currently only concerned with the viewing part, so that will be covered in most depth.

2.1 RTI Theory and Workflows

The RTI workflow is a three step one (see Figure 1) in our context (with the hunt for the physical object excluded). The object is placed central under some capturing device (like Figure 2). A camera is statically mounted on top. And the lights are setup to be lit in order to have a different light position for all captured images. From these images the RTI coefficients are calculated (see Figure 3).

For example the PTM LRGB format[9] stores 9 coefficients per texel: $a_0 - a_5$ for calculating the varying luminance and R, G, B for colours, the resulting

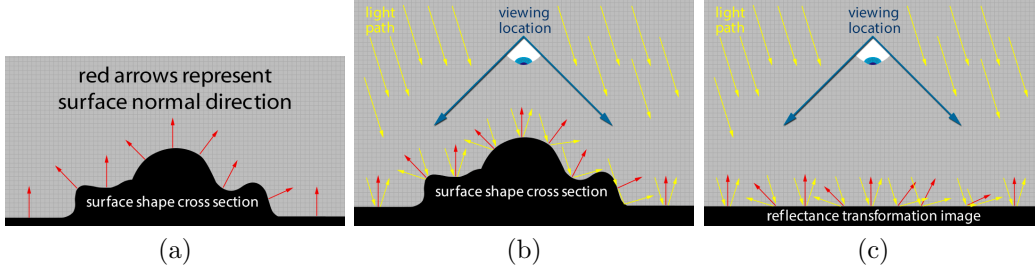


Figure 3: (a) shows the to be captured object’s normals. (b) shows the object with a single viewpoint and light source (in the capturing dome the eye would be camera). (c) visualizes the RTI viewing process. Images from CHI[4].

rendering is calculated in the following way per pixel:

$$\begin{aligned} R(u, v) &= L(u, v)R_n(u, v); \\ G(u, v) &= L(u, v)G_n(u, v); \\ B(u, v) &= L(u, v)B_n(u, v); \end{aligned}$$

With u and v being the texture coordinates. R_n , G_n and B_n being the raw unscaled input colours and $L(u, v)$ calculated as following:

$$L(u, v, l_u, l_v) = a_0(u, v)l_u^2 + a_1(u, v)l_v^2 + a_2(u, v)l_u l_v + a_3(u, v)l_u + a_4(u, v)l_v + a_5(u, v)$$

Where a_0 - a_5 are the previously fit coefficients and l_u and l_v are x and y components of the light vector projected into the texture plane. The coefficients are calculated as part of the RTIBuilder process, for details refer to [9]. Other file formats will have slightly different sets of coefficients, but the same principles hold, e.g. PTM RGB has 6 luminance coefficients per colour channel and no fixed colours.

2.2 Fileformats

The most comprehensive overview on the current state of the art is done by the American Library of Congress (LoC) as part of its digital preservation effort, with sections on the ptm[1] and rti[2] formats. The current PTM specification by Malzbender et al[9] from 2001 specifies 9 different formats, of which “the most commonly used encoding formats are [...] PTM_FORMAT_LRGB (aka LRGB PTM) and PTM_FORMAT_RGB (aka RGB PTM)”[1], so most effort was spent to support these inside the oxrti implementation. Both formats begin with a metadata block consisting of 4 elements separated by newlines:

- The version prefix, which should be ‘PTM.1.2’ for recent files.
- The format string, e.g. PTM_FORMAT_LRGB.

- Image size, the dimensions in pixels.
- Scales and biases, to scale and bias the coefficients before calculation.

This metadata block is followed by the raw texel data in forms depending on the format. All in reversed scanline order. PTM_FORMAT_RGB stores a_0 - a_5 per texel in separated RGB blocks. PTM_FORMAT_LRGB stores a_0 - a_5 per texel first, then followed by RGB blocks of a single coefficient per texel. The destructuring of these is presented in detail in ???. Different embedded JPEG compression styles are specified as well, but due to their relatively small popularity and increased complexity skipped over for the initial oxrti version.

The RTI fileformat specification by Corsini et al.[3] is a newer format supporting Hemi-Spherical Harmonics (HSH) in addition to PTM’s biquadratic polynomials, as HSH “handles shiny surfaces and specular highlights (bright spots of light appearing on shiny objects) better than PTM, which yields matte images.”[2]. One additional difference is the support of metadata as an XMP packet at the end of the file. As most locally available data is in PTM format, support for the RTI format is initially skipped for this project, but it will be easy to add support in the future.

2.3 RTI Viewers

Following RTI Viewers are currently available:

PTM Viewer from HP. The initial viewer application, available at [6]. Downloads for Windows, MacOS, HP-UX and Linux. Only rudimentary functionality of changing the light direction. The MacOS version did not run on the author’s computer. No source code available.

RTIViewer available from the CHI at [5]. Most widely used viewer. Downloads for Windows and MacOS. Source code only available per request: “This software is available under the Gnu General Public License version 3. If you wish to receive a copy of the source code, please send email to info@c-h-i.org.” PTM and RTI formats supported. Features: Light control, multiple rendering modes, bookmarks.

InscriptiFact Viewer by the University of Southern California. Offered in two flavours, a downloadable Java program[7] and a embedded Java applet[8] (not working as of August 27, 2018). No source code available. Features: Light control, multiple rendering modes.

webRTIViewer by the Visual Computing Laboratory. Download at [11]. Support for a proprietary format, requiring pre-processing with a C++ command line utility. Source code included in the download. Features: Light control.

OxRTI SVG Viewer by Christopher Ramsey. SVG based web viewer available at [10], developed at the same time as this project. Requiring pre-processed PTMs. No source code available. Features: Light control, multiple rendering modes, comments and painting.

This is the specification of the BTF fileformat, as of version 1.0 on August 27, 2018. It was developed co-supervisor Stefano Gogioso, with input from and extension by the author of the enclosing thesis in relation to the oxrti viewer.

A BTF File Format

This section describes the BTF file format. The aim of this file format is to provide a generic container for BTF data to be specified using a variety of common formats. Files shall have the `.btf.zip` extension.

A.1 File Structure

A BTF file is a ZIP file containing the following:

- A **manifest** file in JSON format, named `manifest.json`. The manifest contains all information about the BRDF/BSDF model being used, including the names for the available **channels** (e.g. R, G and B for the 3-channel RGB), the names of the necessary **coefficients** (e.g. bi-quadratic coefficients) and the **image file format** for each channel.
- A single folder named `data`, with sub-folders having names in 1-to-1 correspondence with the channels specified in the manifest.
- Within each channel folder, greyscale image files having names in 1-to-1 correspondence with the coefficients specified in the manifest, each in the image file format specified in the manifest for the corresponding channel. For example, if one is working with RGB format (3-channels named R, G and B) in the PTM model (five coefficients `a2`, `b2`, `a1`, `b1` and `c`, specifying a bi-quadratic) using 16-bit greyscale bitmaps, the file `/data/B/a2.bmp` is the texture encoding the `a2` coefficient for the blue channel of each point in texture space.
- The datafiles are all in reversed scanline order (meaning from bottom to top), to keep aligned with the original PTM format and allow easier loading into WebGL.

In case of usage with the oxrti viewer, following files can be present in addition to those mentioned above:

A.2 Manifest

The manifest for the BTF file format is a JSON file with root dictionary. The **root** element has two mandatory child elements: one named **data**, and one named **name** with the option of additional child elements (with different names) left open to future extensions of the format.

- The **name** element is a string with a name of the contained object.
- The **data** element has for entries, named **width**, **height**, **channels** and **channel-model**. The **width** and **height** attributes have values in the positive integers describing the dimensions of the BTDF. The **channel-model** attribute has value a non-empty alphanumeric string uniquely identifying the BRDF/BSDF colour model used by the BTF file (see Options section below). The **channels** element has an arbitrary amount of named **channel** entries, according to the **channel-model**.
- Additionally the **data** element has one untyped entry named **formatExtra**, where format implementation specific data can be stored.
- Each **channel** has an **coefficients** child consisting of an arbitrary number of **coefficient** entries, as well as one **coefficient-model** attribute. The **coefficient-model** attribute has value a non-empty alphanumeric string uniquely identifying the BRDF/BSDF approximation model used by the BTF file (see Options section below).
- Each **coefficient** element has one attribute: **format**. The **format** attribute has value a non-empty alphanumeric string uniquely identifying the image file format used to store the channel values (see Options section below).

A.3 Textures

Each image file `/data/CHAN/COEFF.EXT` has the same dimensions specified by the **width** and **height** attributes of the **data** element in the manifest, and is encoded in the greyscale image file format specified by the **format** attribute of the unique **coefficient** element with attribute **name** taking the value **COEFF** (the extension `.EXT` is ignored). The colour value of a pixel (**u,v**) in the image is the value for coefficient **COEFF** of channel **CHAN** in the BRDF/BSDF for point (**u,v**), according to the model jointly specified by the values of the attribute **model** for element **channels** (colour model) and the attribute **model** for element **coefficients** (approximation model).

A.4 Options

At present, the following values are defined for attribute `channel-model` of element `channels`.

- **RGB**: the 3-channel RGB colour model, with channels named **R**, **G** and **B**. This colour model is currently under implementation.
- **LRGB**: the 4-channel LRGB colour model, with channels named **L**, **R**, **G** and **B**. This colour model is currently under implementation.
- **SPECTRAL**: the spectral radiance model, with an arbitrary non-zero number of channels named either all by wavelength (format `---nm`, with `---` an arbitrary non-zero number) or all by frequency format `---Hz`, with `---` an arbitrary non-zero number. This colour model is planned for future implementation.

At present, the following values are defined for attribute `model` of element `coefficients`, where the ending character `*` is to be replaced by an arbitrary number greater than or equal to 1.

- **flat**: flat approximation model (no dependence on light position). This approximation model is currently under implementation.
- **RTIpoly***: order-`*` polynomial approximation model for RTI (single view-point BRDF). This approximation model is currently under implementation.
- **RTIharmonic***: order-`*` hemispherical harmonic approximation model for RTI (single view-point BRDF). This approximation model is currently under implementation.
- **BRDFpoly***: order-`*` polynomial approximation model for BRDFs. This approximation model is planned for future implementation.
- **BRDFharmonic***: order-`*` hemispherical harmonic approximation model for BRDFs. This approximation model is planned for future implementation.
- **BSDFpoly***: order-`*` polynomial approximation model for BSDFs. This approximation model is planned for future implementation.
- **BSDFharmonic***: order-`*` spherical harmonic approximation model for BSDFs. This approximation model is planned for future implementation.

At present, the following values are defined for attribute `format` of elements tagged `coefficient`, where the ending character `*` is the bit-depth, to be replaced by an allowed positive multiple of 8.

- **BMP***: greyscale BMP file format with the specified bit-depth (8, 16, 24 or 32). Support for this format is currently under implementation.

- `PNG*`: PNG file format encoding the specified bit-depth (8, 16, 24, 32, 48 or 64). Support for this format is currently under implementation. Different PNG colour options are used to support different bit-depths:
- `Grayscale` with 8-bit/channel to encode 8-bit bit-depth.
- `Grayscale` with 16-bit/channel to encode 16-bit bit-depth.
- `Truecolor` with 8-bit/channel to encode 24-bit bit-depth.
- `Truecolor` and `alpha` with 8-bit/channel to encode 32-bit bit-depth.
- `Truecolor` with 16-bit/channel to encode 48-bit bit-depth.
- `Truecolor` and `alpha` with 16-bit/channel to encode 64-bit bit-depth.

References

- [1] Library of Congress. *Polynomial Texture Map (PTM) File Format*. June 14, 2018. URL: <https://www.loc.gov/preservation/digital/formats//fdd/fdd000487.shtml> (visited on 08/10/2018).
- [2] Library of Congress. *Reflectance Transformation Imaging (RTI) File Format*. June 9, 2018. URL: <https://www.loc.gov/preservation/digital/formats//fdd/fdd000486.shtml#notes> (visited on 08/10/2018).
- [3] Massimiliano Corsini, Prabath Gunawardane, and Carla Schroer. “RTI Format – Draft 0.9”. In: *Cultural Heritage Imaging Forum* (2010). URL: http://forums.culturalheritageimaging.org/index.php?app=core&module=attach§ion=attach&attach_id=81.
- [4] *Cultural Heritage Imaging | Reflectance Transformation Imaging (RTI)*. URL: <http://culturalheritageimaging.org/Technologies/RTI/> (visited on 08/24/2018).
- [5] *Cultural Heritage Imaging | View: RTIViewer Download*. URL: http://culturalheritageimaging.org/What_We_Offer/Downloads/View/ (visited on 08/26/2018).
- [6] *HP Labs : Research : Polynomial texture mapping : Downloads*. URL: <http://www.hpl.hp.com/research/ptm/downloads/download.html> (visited on 08/27/2018).
- [7] *InscriptiFact*. URL: <http://www.inscriptifact.com/index.shtml> (visited on 08/27/2018).
- [8] *InscriptiFact :: Instructions ::* URL: <http://ruth.usc.edu:7060/index.jsp> (visited on 08/27/2018).
- [9] Thomas Malzbender, Dan Gelb, and Hans Wolters. “Polynomial texture maps”. In: *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*. Vol. 2001. Jan. 1, 2001, pp. 519–528. DOI: 10.1145/383259.383320.
- [10] *OxRTI SVG Viewer*. URL: <https://c14.arch.ox.ac.uk/oxrti/OxRTIViewer.html?file=https://c14.arch.ox.ac.uk/oxrti/example/info.json> (visited on 08/27/2018).
- [11] *Reflectance Transformation Imaging - WebRTIViewer*. URL: <http://vcg.isti.cnr.it/rti/webviewer.php> (visited on 08/27/2018).