Augmented Reality Library for the Web (tracking.js)

by

Eduardo A. Lundgren Melo

Submitted to the Center for Informatics
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

FEDERAL UNIVERSITY OF PERNAMBUCO

February 2013

Author . . . .
Center for Informatics
Mar 20, 2013

Certified by . . . .
Silvio de Barros Melo
Associate Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Master Theses

Augmented Reality Library for the Web (tracking.js)

by

Eduardo A. Lundgren Melo

Submitted to the Center for Informatics
on Mar 20, 2013, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

In this thesis, I designed and implemented an Augmented Reality (AR) framework aiming to provide a common infrastructure to develop applications and to accelerate the use of those techniques on the web in commercial products. It runs on native web browsers without requiring third-party plugins installation.This involves the use of several modern browser specifications as well as implementation of different computer vision algorithms and techniques into the browser environment. These algorithms can be used to detect and recognize faces, identify objects, track moving objects, etc. The source language of the framework is JavaScript that is the language interpreted by all modern browsers. The majority of interpreted languages have limited computational power when compared to compiled languages, such as C. The computational complexity involved in AR requires highly optimized implementations. Some optimizations are discussed and implemented on this work in order to achieve good results when compared with similar implementations in compiled languages. A series of evaluation tests were made, to determine how effective these techniques were on the web.

Thesis Supervisor: Silvio de Barros Melo
Title: Associate Professor

Acknowledgments

This is the acknowledgements section. You should replace this with your own acknowledgements [4] foo [4].

This master thesis has been examined by a Committee as follows:

Professor Silvio de Barros Melo . . . .

Thesis Supervisor
Associate Professor

Professor Veronica Teichrieb . . . .

Chairman, Thesis Committee
Associate Professor

Professor Alvo Dumbledore . . . .

Member, Thesis Committee
Hogwarts School of Witchcraft and Wizardry

Contents

List of Figures

List of Tables

List of Acronyms

AJAX Asynchronous JavaScript and XML

BAST Bug Report Analysis and Search Tool

BTT Bug Report Tracker Tool

BRN Bug Report Network

CCB Change Control Board

Listings

Chapter 1

Introduction

This section introduces this master thesis. It will briefly describe the motivation of the work itself, state the problem that we will focus on solving and shortly discuss the proposed solution. In the end, explain the structure of the next chapters.

Micro-optimization is a technique to reduce the overall operation count of floating point operations. In a standard floating point unit, floating point operations are fairly high level, such as "multiply" and "add"; in a micro floating point unit (μFPU), these have been broken down into their constituent low-level floating point operations on the mantissas and exponents of the floating point numbers.

Chapter two describes the architecture of the μFPU unit, and the motivations for the design decisions made.

Chapter three describes the design of the compiler, as well as how the optimizations discussed in section 1 were implemented.

Chapter four describes the purpose of test code that was compiled, and which statistics were gathered by running it through the simulator. The purpose is to measure what effect the micro-optimizations had, compared to unoptimized code. Possible future expansions to the project are also discussed.

1.1 Motivation

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Augmented Reality, Tracking and Detection, Web, Tracking on the Web

1.2 Problem Definition

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1.3 Objectives

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1.3.1 Augmented Reality Library for the Web

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1.4 Thesis Structure

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Chapter 2

Basic Concepts

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.1 Web

Using concepts from existing hypertext systems, Tim Berners-Lee, computer scientist and at that time employee of CERN, wrote a proposal in March 1989 for what would eventually become the World Wide Web (WWW) [16].

The World Wide Web is a shared information system operating on top of the Internet. Web browsers retrieve content and display from remote web servers using a stateless and anonymous protocol called HyperText Transfer Protocol (HTTP). Web pages are written using a simple language called HyperText Markup Language (HTML). They may be augmented with other technologies such as Cascading Style Sheets (CSS), which adds additional layout and style information to the page, and JavaScript (JS) language, which allows client-side computation. Browsers typically provide other useful features such as bookmarking, history, password management, and accessibility features to accommodate users with disabilities [4].

In the beginning of the web, plain text and images were the most advanced features available on the browsers. In 1994, the World Wide Web Consortium (W3C) was founded to promote interoperability among web technologies. Companies behind web browser development together with the web community, were able to contribute to the W3C specifications [16]. Today's web is a result of the ongoing efforts of an open web community that helps define these technologies and ensure that they're supported in all web browsers. Those contributions transformed the web in a growing universe of interlinked pages and applications, with videos, photos, interactive content, 3D graphics processed by the Graphics Processing Unit (GPU), and other varieties of features without requiring any third-party plugins installation [7]. The significant reuse of open source components among different browsers and the emergence of extensive web standards have caused the browsers to exhibit "convergent evolution" [4].

2.1.1 State of the Art

The browser main functionality is to present a web resource, by requesting it from the server and displaying it on the browser window. There are four major browsers used today: Internet Explorer, Firefox, Safari and Chrome. Currently, the usage share of Firefox, Safari and Chrome together is nearly 60% [17].

Three mature browser implementations were selected and, for each browser, a conceptual architecture was described based on domain knowledge and available documentation. Firefox version 16.0, Safari version 6.0.4 and Chrome version 25.0.1364 were used to derive the reference architecture because they are mature systems, have reasonably large developer communities and user bases, provide good support for web standards, and are entirely open source.

The reference architecture for web browsers based on three well known open source implementations architecture, is shown in Figure 2-1; it comprises eight major subsystems plus the dependencies between them: (1) the User Interface, this includes the address bar, back and forward buttons, bookmarking menu etc. Every part of the browser display except the main window where you see the requested resource; (2) the Browser Engine, an embeddable component that provides a high-level interface for querying and manipulating the Rendering Engine; (3) the Rendering Engine, which performs parsing and layout for HTML documents, optionally styled with CSS; (4) the Networking subsystem, used for network calls, like HTTP requests. It has platform independent interface and underneath implementations for each platform; (5) the JavaScript Parser, used to parse and execute the JavaScript code; (6) the XML Parser; (7) the UI Backend, which provides drawing and windowing primitives, user interface widgets, and fonts. Underneath it uses the operating system user interface methods; and (8) the Data Persistence subsystem, which stores various data associated with the browsing session on disk, including bookmarks, cookies, and cache [4].

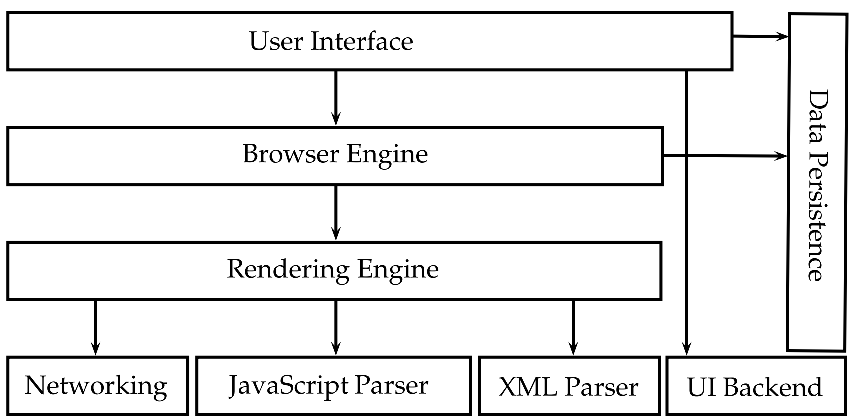

Figure 2-1: Reference architecture for web browsers

Browser subsystem are swappable and could vary for browser vendor, platform or operational system. The browsers mostly differ between different vendors in subsystems (2) the Browser Engine, (3) the Rendering Engine, and (5) the JavaScript Parser. Firefox subsystems (2) and (3) is known as Gecko [15] [13], Safari as WebKit [8] [9] and Chrome uses a fork of WebKit project called Blink [10] [11]. Those browsers subsystems, often called Browser Engines, are shown on Figure 2-2.

Another common swappable subsystem is (5) the JavaScript Parser. JavaScript is a lightweight, interpreted, object-oriented language with first-class functions, most known as the scripting language for Web pages [13]. The JavaScript standard is ECMAScript. As of 2013, all modern browsers fully support ECMAScript 5.1. Older browsers support at least ECMAScript 3 [13] [12].

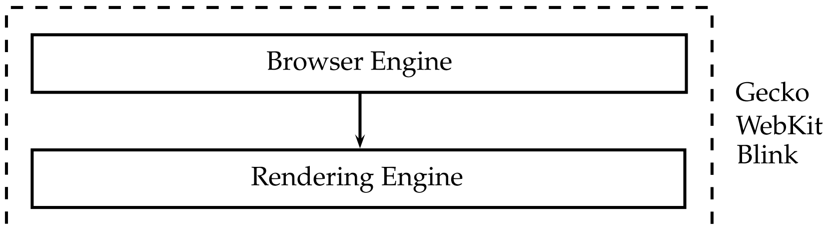

Figure 2-2: Reference architecture for browsers engines

The JavaScript language is intended to be used within some larger environment, be it a browser, server-side scripts, or similar. For a basic example of the language syntax a *println* might have been defined in Listing 2.1.

```
function println(string) {
  window.alert(string);
}
```

Listing 2.1: Basic example of JavaScript syntax

JavaScript core language features comprises few major features: (1) Functions and function scope, function is a "subprogram" that can be called by code external, functions have a scope it references for execution; (2) Global Objects, refer to objects in the global scope, such as general-purpose constructors (*Array, Boolean, Date* etc.), Typed array constructors (*Float32Array, Int32Array, Uint32Array* etc.), Error constructors etc.; (3) Statements, consist of keywords used with the appropriate syntax (*function, if...else, block, break, const, continue, debugger etc.*); (4) Operators and keywords, arithmetic operators, bitwise operators, assignment operators, comparison operators, logical operators, string operators, member operators, conditional operator etc. [14].

As web applications become more and more powerful, adding features such as audio and video manipulation, access to raw data using WebSockets [14], and so forth, it has become clear that there are times when it would be helpful for JavaScript code to be able to quickly and easily manipulate raw binary data. In the past, this had to be simulated by treating the raw data as a string and using the *charCodeAt()* method to read the bytes from the data buffer [14] [6].

However, this is slow and error-prone, due to the need for multiple conversions, especially if the binary data is not actually byte-format data, but, for example, 32-bit integers or floats. Superior, and typed data structures were added to JavaScript specification, such as JavaScript typed arrays [12].

JavaScript typed arrays provide a mechanism for accessing raw binary data much more efficiently. This thesis takes advantage of typed arrays in order to achieve acceptable performance on the web of complex algorithms implementations.

A performance benchmark comparing regular *vs* typed arrays were executed on the three well known open-source browsers, Firefox, Safari and Chrome. The comparison was executed on Mac OS X 10.8.3, 2.6 GHz Intel Core i7 16 GB 1600 MHz RAM. The array types selected were the not strongly typed *Array*; *Float32Array*, represents an array of 32-bit floating point numbers; *Uint8Array*, represents an array of 8-bit unsigned integers.

For each array type a read and a write operation were executed $100,000$ times. In order to not compromise benchmark results caused by run-time type conversion [12], the write value used for each array type were proper selected, e.g. *Number* 1.0 was used for regular arrays *Array*, *Number* 1.0 was used for *Float32Array*, and unsigned *Number* 1 for *Uint8Array*. Regular *vs* typed arrays performance benchmark is shown in Figure 2-3 [1].

As conclusion, typed arrays provides faster read and write operations than regular arrays in JavaScript, i.e. 7872 *ops/sec* for unsigned array *vs* 4437 *ops/sec* for regular arrays in Firefox browser, similar behavior is noticeable on Safari and Chrome, thereby float and unsigned arrays are vastly used on complex algorithms implementations on the web.

Nevertheless, reading and writing raw binary data using typed arrays only solves part of the problem of manipulating video and images data. The other missing

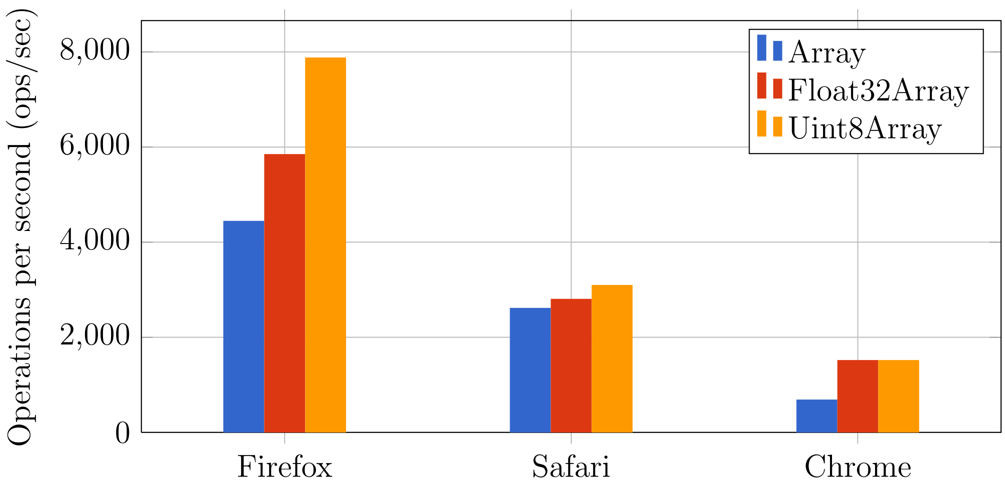

Figure 2-3: Regular *vs* typed arrays performance benchmark

feature was solved by HTML5 *canvas* element, which one important feature is to provide access to the pixel matrix of those medias. The raw binary data is used by Augmented Reality (AR) algorithms.

The *canvas* element provides scripts with a resolution-dependent bitmap canvas, which can be used for rendering graphs, game graphics, art, or other visual images on the fly [2]. Authors should not use the *canvas* element in a document when a more suitable element is available, e.g. it is inappropriate to use a *canvas* element to render a page heading. The usage of *canvas* conveys essentially the same function or purpose as the canvas' bitmap. Listing 2.2 shows an example of a basic *canvas* element HTML markup given a width and height in pixels.

```
<canvas width="200" height="200"></canvas>
```

Listing 2.2: The HTML canvas element markup

For each *canvas* element a "context" is available, then, from that, the drawing context can be accessed and JavaScript commands can be invoked to draw or read data. Browsers can implement multiple canvas contexts and the different APIs provide the drawing functionality. Most of the major browsers include the 2D canvas context capabilities. Individual vendors have experimented with their own three-dimensional canvas APIs, but none of them have been standardized. The HTML5 specification notes, “A future version of this specification will probably define a 3D context” [2]. Even though 3D context is not available in most part of the major browsers, three-dimensional applications are already being developed based on the 2D canvas context.

Is mandatory the use of the *canvas* element to develop AR applications on the web. Canvas provides APIs to: Draw basic shapes, images, videos frames, Bezier and quadratic curves [5]; Apply transformations, translate, rotate and scale; Read raw pixel data etc.

The canvas is a two-dimensional grid that could be described as a simple computer graphics coordinate system [5]. Normally 1 unit in the grid corresponds to 1 pixel on the canvas. The origin of this grid is positioned in the top left corner coordinate $(0, 0)$. All elements are placed relative to this origin. So the position of the top left corner of the blue square becomes x pixels from the left and y pixels from the top coordinate (x, y). The canvas coordinate space is shown on Figure 2-4 [14].

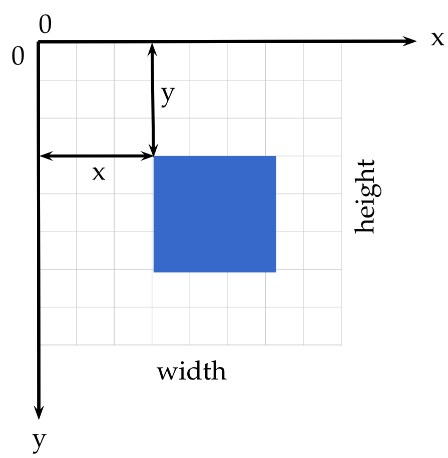

Figure 2-4: The canvas coordinate space

Videos and images can also be “hosted” on a canvas bitmap. Canvas raw binary data can be accessed from the canvas JavaScript API as an object of type *ImageData* [2]. Each object has three properties: width, height and data. The data property is of type *Uint8ClampedArray* that is a one-dimensional array containing the data in RGBA order, as integers in the range 0 to 255. The *Uint8ClampedArray* interface type is specifically used in the definition of the canvas element's 2D API and its structure is similar to the previous shown typed array *Uint8Array* [2] [6].

The *ImageData* data property, or array of pixels, is in row-major order, a multidimensional array in linear memory. For example, consider the 2×3 array $\begin{bmatrix} 1 & 2 & 3 \\ 5 & 5 & 6 \end{bmatrix}$, in row-major order it is laid out contiguously in linear memory as $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$. Each array value is represented as integers between 0 and 255, where each four-integer group represents the four color channels of one pixel: red, group, blue and alpha (RGBA). While RGBA is sometimes described as a color space, it is actually simply a use of the RGB color model [3]. An example of the canvas image data array of pixels is shown on Figure 2-5.

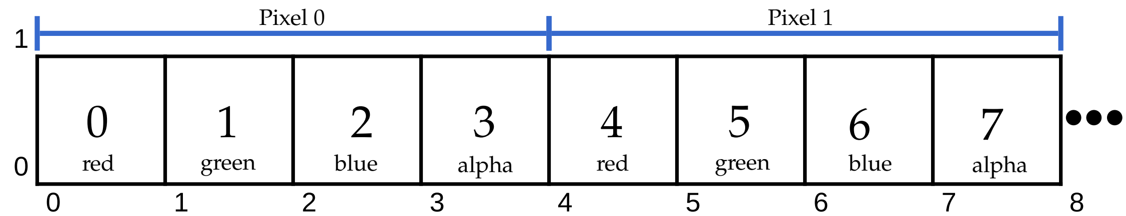

Figure 2-5: The canvas image data array of pixels

2.1.2 Problems of Augmented Reality on the Web

2.2 Augmented Reality

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.2.1 State of the Art

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.3 Tracking and Object Detection

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

2.3.1 State of the Art

Chapter 3

Augmented Reality Library for the Web (tracking.js)

3.1 Introduction

Supported modules: color, keypoints, rapid object detection.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.2 Color Tracking Algorithm

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

3.3 Marker-less Tracking Algorithm (Keypoints)

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

BRIEF, FAST, RANSAC.

3.4 Rapid Object Detection and Tracking Algorithm

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Viola-Jones: Features, Integral images, Learning, Detection, Training a cascade of classifiers, Training data optimization for JavaScript.

Chapter 4

Evaluation

4.1 Tools

OpenCV, JSFlartoolkit, Others. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.2 Scenario Description

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3 Evaluation Methodology

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3.1 Matching Robustness

4.3.2 Oclusion Robustness

4.3.3 FPS

4.4 Results

Graphics, Analysys. Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Chapter 5

Conclusion

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.1 Contributions

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.2 Future Work

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Bibliography

[1] Eduardo A Lundgren Melo. Typed Arrays Performance, 2013.

[2] WHATWG Community. The canvas element, 2013.

[3] Rafael C Gonzalez and Richard E Woods. *Digital Image Processing (3rd Edition)*. Prentice Hall, 2007.

[4] Alan Grosskurth and M.W. Godfrey. A reference architecture for Web browsers. In *21st IEEE International Conference on Software Maintenance (ICSM'05)*, pages 661–664. IEEE, 2005.

[5] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*, volume 2. Cambridge University Press, 2004.

[6] David Herman and Kenneth Russell. Typed Array Specification, 2013.

[7] Ian Hickson. HTML 5 Nightly Specification (W3C), 2013.

[8] Apple Inc. Safari Browser, 2013.

[9] Apple Inc. The WebKit Open Source Project, 2013.

[10] Google Inc. Google Chrome Browser, 2010.

[11] Google Inc. Blink, 2013.

[12] Ecma International. ECMA-262 ECMAScript Language Specification. *JavaScript Specification*, 16(December):1–252, 2009.

[13] Inc. Mozilla. Gecko, 2013.

[14] Inc. Mozilla. Mozilla Developer Network, 2013.

[15] Inc. Mozilla. Mozilla Firefox Browser, 2013.

[16] W C. The World Wide Web Consortium (W3C), 2006.

[17] Wikimedia. Wikimedia Traffic Analysis Report - Browsers, 2013.