Tracking Library for the Web

Eduardo A. Lundgren Melo

Universidade Federal de Pernambuco

Centro de Informática UFPE

Master of Science in Computer Science

Silvio de Barros Melo (*Advisor*)

Veronica Teichrieb (*Co-Advisor*)

Outline

1. Introduction
2. Basic concepts
   - Web
   - Visual tracking
   - Visual tracking on the web
3. Tracking library for the web
4. Results and evaluation
   - Color tracking algorithm
   - Rapid object detection (Viola Jones)
   - Markerless tracking algorithm
5. Conclusions

Outline

1 Introduction

2 Basic concepts
- Web
- Visual tracking
- Visual tracking on the web

3 Tracking library for the web

4 Results and evaluation
- Color tracking algorithm
- Rapid object detection (Viola Jones)
- Markerless tracking algorithm

5 Conclusions

Motivation

- The web browser environment is evolving fast

Motivation

- The web browser environment is evolving fast
- Phones and notebooks devices have embedded web browser

Motivation

- The web browser environment is evolving fast
- Phones and notebooks devices have embedded web browser
- Entertainment solutions are gaining space on the web

Motivation

- The web browser environment is evolving fast
- Phones and notebooks devices have embedded web browser
- Entertainment solutions are gaining space on the web
- Vision is an accurate and low-cost solution

Problem definition

- Visual tracking requires video capturing and processing

Problem definition

- Visual tracking requires video capturing and processing
- Video processing requires high computational complexity

Problem definition

- Visual tracking requires video capturing and processing
- Video processing requires high computational complexity
- JavaScript is a language interpreted by all web browsers

Problem definition

- Visual tracking requires video capturing and processing
- Video processing requires high computational complexity
- JavaScript is a language interpreted by all web browsers
- Interpreted languages have limited computational power

Objectives

- Facilitate user interaction with the web browser

Objectives

- Facilitate user interaction with the web browser
- Accelerate the use of visual tracking in commercial products

Objectives

- Facilitate user interaction with the web browser
- Accelerate the use of visual tracking in commercial products
- Implement a cross-platform tracking library for the web

Outline

1 Introduction

2 Basic concepts
- Web
- Visual tracking
- Visual tracking on the web

3 Tracking library for the web

4 Results and evaluation
- Color tracking algorithm
- Rapid object detection (Viola Jones)
- Markerless tracking algorithm

5 Conclusions

The beginning of the web

The beginning of the web

- Plain text and images were the most advanced features

The beginning of the web

- Plain text and images were the most advanced features
- In 1994, the World Wide Web Consortium (W3C) was founded

The beginning of the web

- Plain text and images were the most advanced features
- In 1994, the World Wide Web Consortium (W3C) was founded
- Companies were able to contribute to the W3C specifications

The beginning of the web

- Plain text and images were the most advanced features
- In 1994, the World Wide Web Consortium (W3C) was founded
- Companies were able to contribute to the W3C specifications
- Today's web is a result of the ongoing efforts of an open web

The modern web

Browser technologies

Browser architecture

Browser architecture

Visual tracking

Visual tracking

Visual tracking workflow on the web

1. Request user web-cam access

1. Request user web-cam access

2. Capture web-cam stream

2. Capture web-cam stream

```
<script>
  navigator.getUserMedia({ video: true }, function(localMediaStream) {
    // Stream captured
  }, onFail);
</script>
```

2. Capture web-cam stream

3. Reproduce web-cam stream into the video

3. Reproduce web-cam stream into the video

```
<video autoplay></video>
<script>
  var video = document.querySelector('video');
  navigator.getUserMedia({video: true}, function(localMediaStream) {
      video.src = window.URL.createObjectURL(localMediaStream);
      video.onloadedmetadata = function(e) { alert('Ready to go.') };
  }, onFail);
</script>
```

3. Reproduce web-cam stream into the video

```
<video autoplay></video>
<script>
  var video = document.querySelector('video');
  navigator.getUserMedia({video: true}, function(localMediaStream) {
      video.src = window.URL.createObjectURL(localMediaStream);
      video.onloadedmetadata = function(e) { alert('Ready to go.') };
  }, onFail);
</script>
```

3. Reproduce web-cam stream into the video

```
<video autoplay></video>
<script>
  var video = document.querySelector('video');
  navigator.getUserMedia({video: true}, function(localMediaStream) {
      video.src = window.URL.createObjectURL(localMediaStream);
      video.onloadedmetadata = function(e) { alert('Ready to go.') };
  }, onFail);
</script>
```

3. Reproduce web-cam stream into the video

4. Process video data using canvas

4. Process video data using canvas

5. Access canvas data using JavaScript typed arrays

- In the past, raw data was accessed as a string

5. Access canvas data using JavaScript typed arrays

- In the past, raw data was accessed as a string
- Browsers needed a quick way to manipulate raw binary data

5. Access canvas data using JavaScript typed arrays

- In the past, raw data was accessed as a string
- Browsers needed a quick way to manipulate raw binary data
- Typed data structures were added to JavaScript

5. Access canvas data using JavaScript typed arrays

- In the past, raw data was accessed as a string
- Browsers needed a quick way to manipulate raw binary data
- Typed data structures were added to JavaScript
- JavaScript-typed arrays access raw binary more efficiently

5. Access canvas data using JavaScript typed arrays

5. Access canvas data using JavaScript typed arrays

What is the relation between typed arrays and canvas?

- Videos and images pixels can be drawn on a canvas bitmap

What is the relation between typed arrays and canvas?

- Videos and images pixels can be drawn on a canvas bitmap
- Canvas raw binary data can be accessed from JavaScript

What is the relation between typed arrays and canvas?

- Videos and images pixels can be drawn on a canvas bitmap
- Canvas raw binary data can be accessed from JavaScript
- Canvas array of pixels, is in row-major order

What is the relation between typed arrays and canvas?

- Videos and images pixels can be drawn on a canvas bitmap
- Canvas raw binary data can be accessed from JavaScript
- Canvas array of pixels, is in row-major order
- Consider the 2×3 array $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, in row-major order it is laid out contiguously in linear memory as $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \end{bmatrix}$.

What is the relation between typed arrays and canvas?

Outline

1 Introduction

2 Basic concepts
- Web
- Visual tracking
- Visual tracking on the web

3 Tracking library for the web

4 Results and evaluation
- Color tracking algorithm
- Rapid object detection (Viola Jones)
- Markerless tracking algorithm

5 Conclusions

Related work

- FLARToolKit: a port of ARToolKit marker tracking library to ActionScript

Related work

- JSARToolkit: is a JavaScript port of FLARToolKit

Related work

- Unifeye Viewer: a robust markerless tracking solution for the web to ActionScript

Flash vs HTML5

tracking.js

tracking.js
About
Examples
Download now
Fork on Github
tracking.js
Change the way you interact with your browser
01:36
vimeo
Download now
Fork on Github
Example
Basic Usage
This example is a simple way to initialize the user browser camera and start tracking for objects with color magenta.
There are two available callbacks, onFound is fired when the object is detected and onNotFound does the opposite.
The onFound callback receives as argument a track
var videoCamera = new tracking.VideoCamera().render().renderVideoCanvas();
videoCamera.track({
type: 'color',
color: 'magenta',
onFound: function(track) {
console.log(track.x, track.y, track.z);
},
onNotFound: function() {}

tracking.js

Tracking library for the web

Common infrastructure to develop visual tracking applications and to accelerate the use of those techniques on the web in commercial products.

Library features

1. Color tracking

Library features

2. Rapid object detection (Viola Jones)

Library features

3. Markerless tracking algorithm

Library modules - Base classes

Library modules - Visual tracking classes

Color tracking algorithm

Figure : © http://www.flickr.com/photos/laynecom/8674644879/

Color tracking algorithm

```
var videoCamera = new tracking.VideoCamera();
videoCamera.track({
    type: 'color',
    color: 'magenta',
    onFound: function(track) {
      // do your logic here.
    }
});
```

Color tracking algorithm - Color difference evaluation

$$\|C_1 - C_2\| = \sqrt{(C_{1,R} - C_{2,R})^2 + (C_{1,G} - C_{2,G})^2 + (C_{1,B} - C_{2,B})^2}$$

Color tracking algorithm - Color blob detection

Rapid object detection (Viola Jones)

Rapid object detection (Viola Jones)

```
var videoCamera = new tracking.VideoCamera();
videoCamera.track({
    type: 'human',
    data: 'frontal_face',
    onFound: function(track) {
      // do your logic here.
    }
});
```

Rapid object detection (Viola Jones)

- Robust and extremely rapid object detection

Rapid object detection (Viola Jones)

- Robust and extremely rapid object detection
- Became popular mainly because rapid face detection

Rapid object detection (Viola Jones)

- Robust and extremely rapid object detection
- Became popular mainly because rapid face detection
- A training phase is required

Rapid object detection (Viola Jones)

- Robust and extremely rapid object detection
- Became popular mainly because rapid face detection
- A training phase is required
- A scanning detector is what makes the detection

Rapid object detection (Viola Jones)

Rapid object detection (Viola Jones)

Integral Image

Rectangle features can be computed very rapidly using an intermediate representation for the image which we call the integral image.

The integral image at location x, y contains the sum of the pixels above and to the left of x, y, inclusive

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

Rapid object detection (Viola Jones)

Rapid object detection (Viola Jones)

Scanning detector algorithm

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image
3. For each block location, loop the tree and evaluate each stage

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image
3. For each block location, loop the tree and evaluate each stage
4. Positive stage evaluate next stage, otherwise stops the loop

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image
3. For each block location, loop the tree and evaluate each stage
4. Positive stage evaluate next stage, otherwise stops the loop
5. If all stages were positive store the rectangle

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image
3. For each block location, loop the tree and evaluate each stage
4. Positive stage evaluate next stage, otherwise stops the loop
5. If all stages were positive store the rectangle
6. Once the tree is done, group the overlapping rectangles

Rapid object detection (Viola Jones)

Scanning detector algorithm

1. Create or scale a 20×20 squared block by 1.25 per iteration
2. Loop the block by Δ pixels over the image
3. For each block location, loop the tree and evaluate each stage
4. Positive stage evaluate next stage, otherwise stops the loop
5. If all stages were positive store the rectangle
6. Once the tree is done, group the overlapping rectangles
7. Find the best rectangle of each the group (merging phase)

Rapid object detection (Viola Jones)

1. Optimization: merging phase

Rectangles are used partitioned into a disjoint set data structure. On this work it was replaced by an alternative called Minimum Neighbor Area Grouping.

Rapid object detection (Viola Jones)

1. Optimization: merging phase

Rectangles are used partitioned into a disjoint set data structure. On this work it was replaced by an alternative called Minimum Neighbor Area Grouping.

2. Optimization: training data

OpenCV uses XML to store training data. This work proposes the usage of row-major order typed arrays.

Rapid object detection (Viola Jones)

haarcascade_frontalface_alt.xml

```
<haarcascade_frontalface_alt type_id="opencv-haar-classifier">
  <size>20 20</size>
  <stages>
    <_>
      <!-- stage 0 -->
      <trees>
        <_>
          <!-- tree 0 -->
          <_>
            <!-- root node -->
            <feature>
              <rects>
                <_>3 7 14 4 -1.</_>
                <_>3 9 14 2 2.</_></rects>
              <tilted>0</tilted></feature>
            <threshold>4.0141958743333817e-003</threshold>
            <left_val>0.0337941907346249</left_val>
            <right_val>0.8378106951713562</right_val></_></_>
        <_>
          <!-- tree 1 -->
          <_>
            <!-- root node -->
```

frontal_face.js

```
[[-1,0.822689414024353,[[3,7,14,4,-1,3,9,14,2,2,0.004014195874333382,0.0337941907346249,
0.8378106951713562],[1,2,18,4,-1,7,2,6,4,3,0.0151513395830989,0.1514132022857666,0.
7488812208175659],[1,7,15,9,-1,1,10,15,3,3,0.004210993181914091,0.0900492817163467,0
.6374819874763489]]],[0,6.956608772277832,[[5,6,2,6,-1,5,9,2,3,2,0.
0016227109590545297,0.0693085864186287,0.7110946178436279],[7,5,6,3,-1,9,5,2,3,3,0.
0022906649393209819,0.1795803010463715,0.6668692231178284],[4,0,12,9,-1,4,3,12,3,3,0.
0050025708042085 17,0.1693672984838486,0.6554006934165955],[6,9,10,8,-1,6,13,10,4,2,0
.007965989410877228,0.5866332054138184,0.0914145186543465],[3,6,14,8,-1,3,10,14,4,2,
-0.003522701095789671,0.1413166970014572,0.6031895875930786],[14,1,6,10,-1,14,1,3,10
,2,0.0366676896810532,0.3675672113895416,0.7920318245887756],[7,8,5,12,-1,7,12,5,4,3
,0.009336147457361221,0.6161385774612427,0.2088509947061539],[1,1,18,3,-1,7,1,6,3,3,
0.008696131408214569,0.2836230993270874,0.6360273957252502],[1,8,17,2,-1,1,9,17,1,2,
0.0011488880263641477,0.2223580926656723,0.5800700783729553],[16,6,4,2,-1,16,7,4,1,2
,-0.002148468978703022,0.2406464070081711,0.5787054896354675],[5,17,2,2,-1,5,18,2,1,
2,0.002121906029060483,0.5559654831886292,0.136223703622818],[14,2,6,12,-1,14,2,3,12
,2,-0.0939491465687752,0.8502737283706665,0.4717740118503571],[4,0,4,12,-1,4,0,2,6,2
,6,6,2,6,2,0.0013777789426967502,0.5993673801422119,0.2834529876708984],[2,11,18,8,-
1,8,11,6,8,3,0.0730631574988365,0.4341886043548584,0.7060034275054932],[5,7,10,2,-1,
5,8,10,1,2,0.0003676738997455 6863,0.3027887940406799,0.6051574945449829],[15,11,5,3,
-1,15,12,5,1,3,-0.0060479710809886456,0.17984339594841,0.5675256848335266]]],[1,9.
498542785644531,[[5,3,10,9,-1,5,6,10,3,3,-0.0165106896311045,0.6644225120544434,0.
1424857974052429],[9,4,2,14,-1,9,11,2,7,2,0.002705249935388565,0.6325352191925049,0.
```

Feature detector (FAST)

Feature detector (FAST)

Feature detector (FAST)

1. Optimization: no machine learning step

The steps required for machine learning involves extra math operations, such as *log*. The results were good without it.

Feature detector (FAST)

1. Optimization: no machine learning step

The steps required for machine learning involves extra math operations, such as *log*. The results were good without it.

2. Optimization: no non-maximal suppression step

To remove corners which have an adjacent corner requires extra access to the typed array.

Feature extractor (BRIEF)

Feature extractor (BRIEF)

To generate the binary strings it is defined the test τ on patch $\mathbf{p}$ of size $\mathbf{S} \times \mathbf{S}$ as:

$$\tau(\mathbf{p}; x, y) := \begin{cases} 1 & \text{if} \quad \mathbf{p(x)} < \mathbf{p(y)}, \\ 0 & \text{otherwise} \end{cases}$$

Feature extractor (BRIEF)

The n_d-dimensional bit-string is our BRIEF descriptor for each keypoint:

$$f_{n_d}(\mathbf{p}) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(\mathbf{p}; x, y).$$

In this work $n_d = 128$ was used. The number of bytes required to store the descriptor can be calculated by $k = n_d/8$.

Feature extractor (BRIEF)

Feature extractor (BRIEF)

The weighted Hamming distance is computed by:

$$WHam(x, y) = \sum_{i=1}^{n} w_i(b_i(x) \otimes b_i(y))$$

$$b_1 = 0000000001...$$

$$b_2 = 0000000011...$$

$$b_1 \otimes b_2 = 0000000010...$$

$$WHam = 1$$

Feature extractor (BRIEF)

Homography estimation

Random sample consensus (RANSAC)

RANSAC (Random Sample Consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers. It is the most commonly used robust estimation method for homographies.

Outline

1 Introduction

2 Basic concepts
- Web
- Visual tracking
- Visual tracking on the web

3 Tracking library for the web

4 Results and evaluation
- Color tracking algorithm
- Rapid object detection (Viola Jones)
- Markerless tracking algorithm

5 Conclusions

Evaluation methodology

1. Examples

Evaluation methodology

1. Examples

2. Performance

Frames per second (FPS) metric. All tests were executed on Google Chrome browser version 28.0.1500.71, Mac OS X 10.8.3, 2.6 GHz Intel Core i7 16 GB 1600 MHz RAM.

Evaluation methodology

1. Examples

2. Performance

Frames per second (FPS) metric. All tests were executed on Google Chrome browser version 28.0.1500.71, Mac OS X 10.8.3, 2.6 GHz Intel Core i7 16 GB 1600 MHz RAM.

3. Partial occlusion robustness

Examples of how each technique behaves under partial occlusion.

Evaluation methodology

README.md

stats.js

JavaScript Performance Monitor

This class provides a simple info box that will help you monitor your code performance.

- **FPS** Frames rendered in the last second. The higher the number the better.
- **MS** Milliseconds needed to render a frame. The lower the number the better.

Screenshots

Examples

Performance

Oclusion robustness

Examples

Examples

Performance

Oclusion robustness

Examples

Pro
JavaScript
Techniques
Pro
JavaScript
Techniques
Pro
JavaScript
Techniques
Pro
JavaScript
Techniques

Performance

Oclusion robustness

Outline

1 Introduction

2 Basic concepts
- Web
- Visual tracking
- Visual tracking on the web

3 Tracking library for the web

4 Results and evaluation
- Color tracking algorithm
- Rapid object detection (Viola Jones)
- Markerless tracking algorithm

5 Conclusions

Benefits of a JavaScript tracking solution

	OSAKit	*tracking.js* + color	*tracking.js* + face
Avg frame rate	25 FPS	30 FPS	25 FPS
Avg file size	22 MB	8 KB	187 KB
Avg load time (256 Kbps)	11 min	0.25 seconds	5 seconds

Contributions

- A tracking library for the web called *tracking.js*

Contributions

- A tracking library for the web called *tracking.js*
- Optimizations for existing techniques in order to run in real-time on the web

Contributions

- A tracking library for the web called *tracking.js*
- Optimizations for existing techniques in order to run in real-time on the web
- Pioneered a library that brings state-of-the-art techniques for computer vision to the web

Contributions

- A tracking library for the web called *tracking.js*
- Optimizations for existing techniques in order to run in real-time on the web
- Pioneered a library that brings state-of-the-art techniques for computer vision to the web
- Academic material about web browser concepts

Future work

- Publication about web browser concepts

Future work

- Publication about web browser concepts
- Publication comparing JavaScript tracking solutions with plugin-based and server-side solutions

Future work

- Publication about web browser concepts
- Publication comparing JavaScript tracking solutions with plugin-based and server-side solutions
- Double Exponential Smoothing technique

Future work

- Publication about web browser concepts
- Publication comparing JavaScript tracking solutions with plugin-based and server-side solutions
- Double Exponential Smoothing technique
- Perspective-n-Point problem (PnP) calculation

Future work

- Publication about web browser concepts
- Publication comparing JavaScript tracking solutions with plugin-based and server-side solutions
- Double Exponential Smoothing technique
- Perspective-n-Point problem (PnP) calculation
- Image processing layer capable of transformations and filters

Website

Figure : http://trackingjs.com

Popularity

Popularity

David Herman
@littlecalculist
Following
Spontaneous applause for Eduardo Lundgren's incredible demos of trackingjs.com #braziljs
Lisa Larson-Kelley
@lisamarienyc
Following
A library I wish I had back in the day with Flash -- #javascript #webcam gestures, tracking tracking.js buff.ly/1cmlNp2
HTML5 Weekly
@HTML5Weekly
Follow
Tracking.js: Webcam/Camera Based Interaction for Web Page Elements - trackingjs.com

Popularity

- 445 tweets mentioning *tracking.js*

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views
- Project hosted on Mozilla Developer Network

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views
- Project hosted on Mozilla Developer Network
- Talk presented on DevFest Brazil - Brazil, 2012

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views
- Project hosted on Mozilla Developer Network
- Talk presented on DevFest Brazil - Brazil, 2012
- Talk presented on W3C Conference - Brazil, 2012

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views
- Project hosted on Mozilla Developer Network
- Talk presented on DevFest Brazil - Brazil, 2012
- Talk presented on W3C Conference - Brazil, 2012
- Talk presented on HTML5 Dev Conf - USA, 2013

Popularity

- 445 tweets mentioning *tracking.js*
- 31,258 website page views
- Project hosted on Mozilla Developer Network
- Talk presented on DevFest Brazil - Brazil, 2012
- Talk presented on W3C Conference - Brazil, 2012
- Talk presented on HTML5 Dev Conf - USA, 2013
- Talk accepted on JS.everywhere - USA, 2013

Thank you!