Tracking Library for the Web

Eduardo A. Lundgren Melo

Universidade Federal de Pernambuco

Centro de Informática U·F·P·E

Master of Science in Computer Science

Silvio de Barros Melo (*Advisor*)

Veronica Teichrieb (*Co-Advisor*)

Outline

Outline

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

The beginning of the web

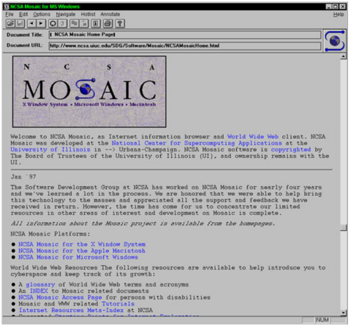


Figure : Mosaic is the web browser credited with popularizing the World Wide Web.

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

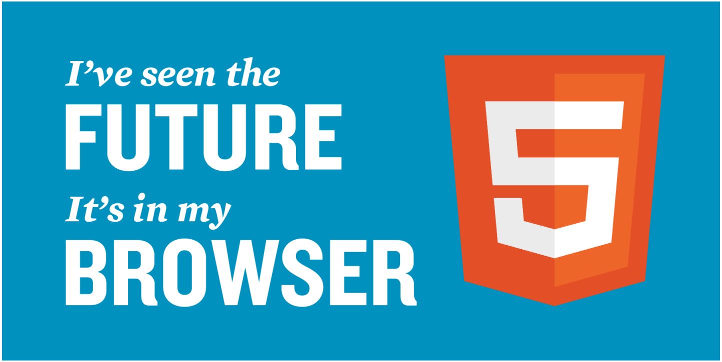
The beginning of the web

- 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



The modern web

- Contributions transformed the web in a growing universe

The modern web

- Contributions transformed the web in a growing universe
- Videos, audio, photos, interactive content, 3D graphics

The modern web

- Contributions transformed the web in a growing universe
- Videos, audio, photos, interactive content, 3D graphics
- Processed by the Graphics Processing Unit (GPU)

The modern web

- Contributions transformed the web in a growing universe
- Videos, audio, photos, interactive content, 3D graphics
- Processed by the Graphics Processing Unit (GPU)
- Without requiring any third-party plugins installation

Browser technologies

Browser architecture

Figure : Web browsers running on different devices.

Browser architecture

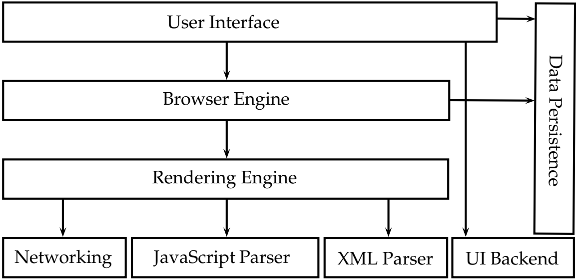


Figure : Reference architecture for web browsers.

Visual tracking

Tracking an object in a video sequence means continuously identifying its location when either the object or the camera are moving.

Figure : Example of an accurate object tracking robust to occlusion.

Visual tracking

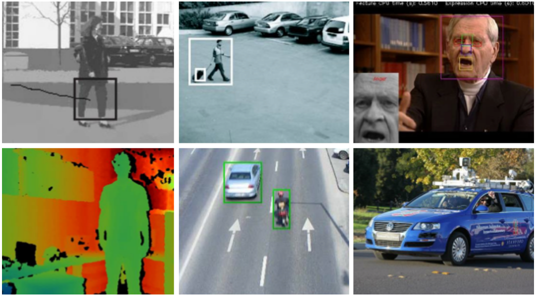
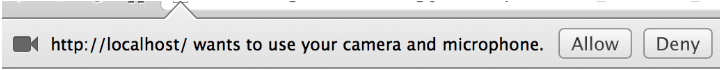


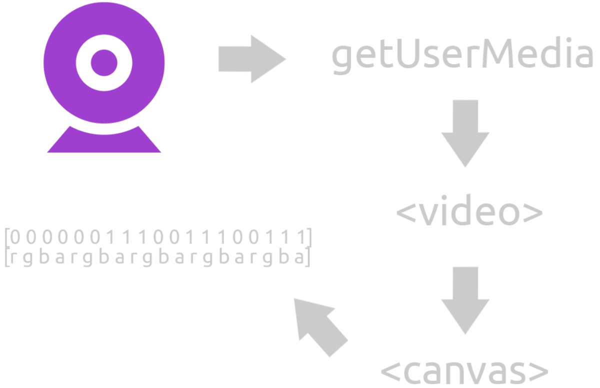
Figure : Computer vision applications: motion-based recognition (top left); automated surveillance (top center); video indexing (top right); human-computer interaction (bottom left); traffic monitoring (bottom center); vehicle navigation (bottom right).

Visual tracking workflow on the web

1. Request user web-cam access



1. Request user web-cam access



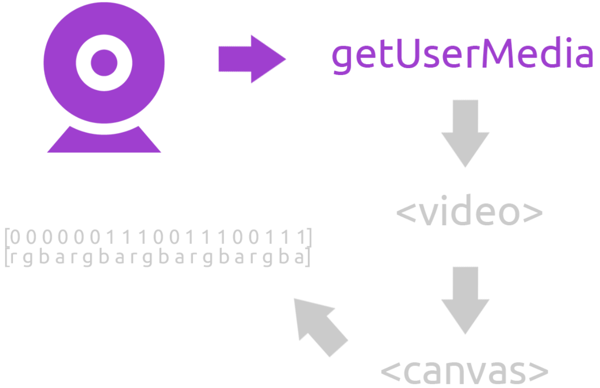
2. Capture web-cam stream

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

Figure : Video and audio HTML5 elements.

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>
```

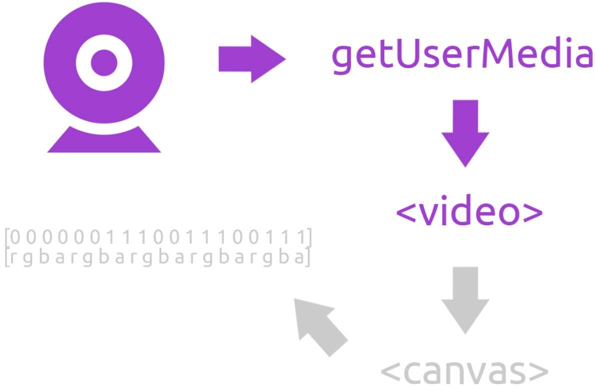
3. Reproduce web-cam stream into the video

```
<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

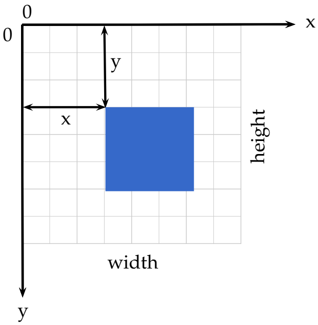


Figure : The canvas element.

4. Process video data using canvas

- Introduced as a new element on HTML5

4. Process video data using canvas

- Introduced as a new element on HTML5
- Resolution-dependent bitmap canvas

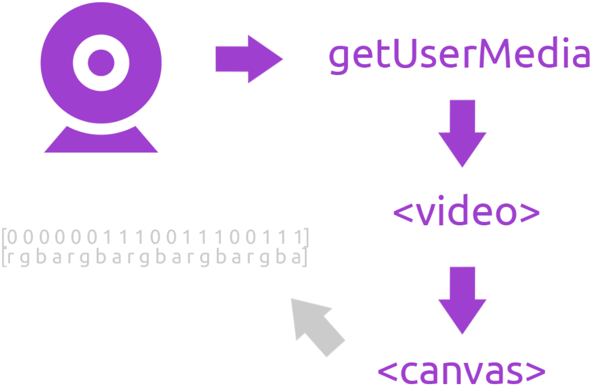
4. Process video data using canvas

- Introduced as a new element on HTML5
- Resolution-dependent bitmap canvas
- Two-dimensional grid, computer graphics coordinate system

4. Process video data using canvas

- Introduced as a new element on HTML5
- Resolution-dependent bitmap canvas
- Two-dimensional grid, computer graphics coordinate system
- Can render images, video frames or shapes

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

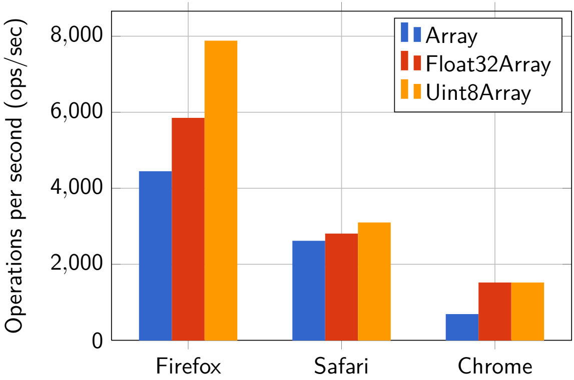
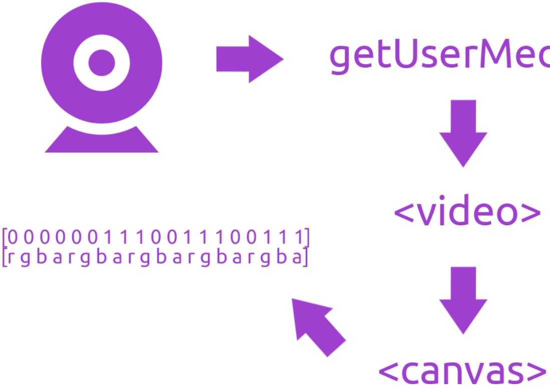


Figure : Regular *vs* typed arrays performance benchmark.

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?

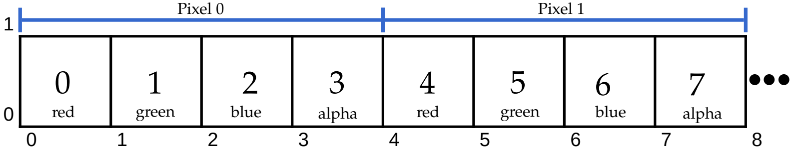
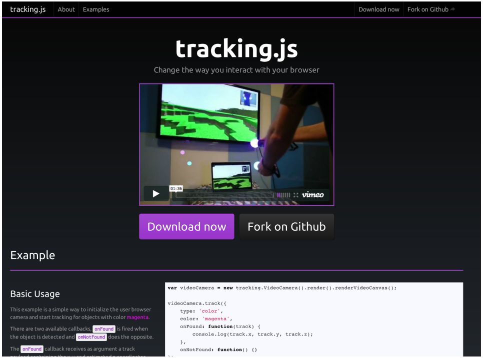


Figure : The canvas image data array of pixels.

Outline

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.

Related work

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

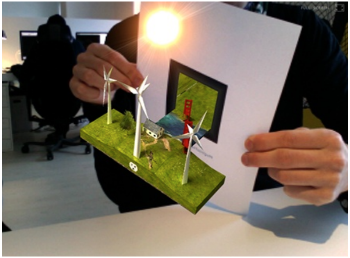


Figure : Marker based AR for the web using FLARToolKit.

Related work

- JSARToolkit: is a JavaScript port of FLARToolKit

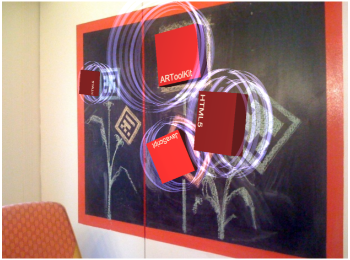


Figure : Marker-based AR for the web using JSARToolKit.

Related work

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

Figure : Markerless example of image projected over a magazine cover.

Flash vs HTML5

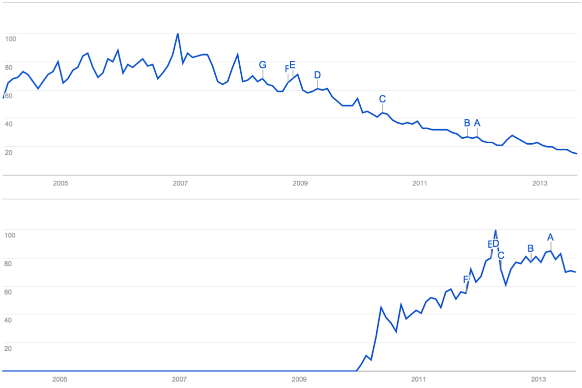


Figure : Google trends results for "flash games" on the top, and for "html5 games" on the bottom.

Library features

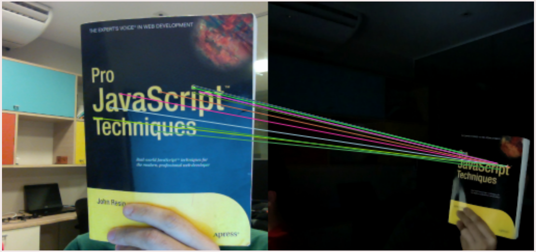
1. Color tracking

Library features

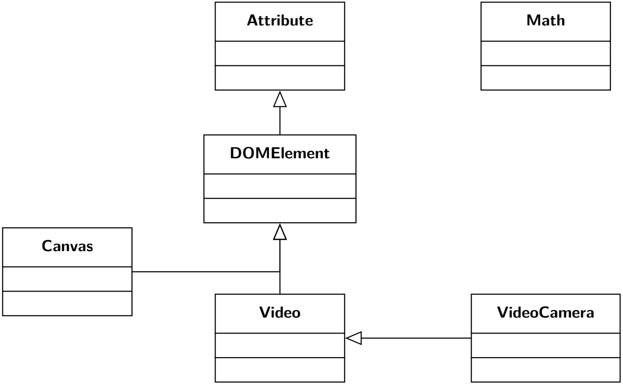
2. Rapid object detection (Viola Jones)

Library features

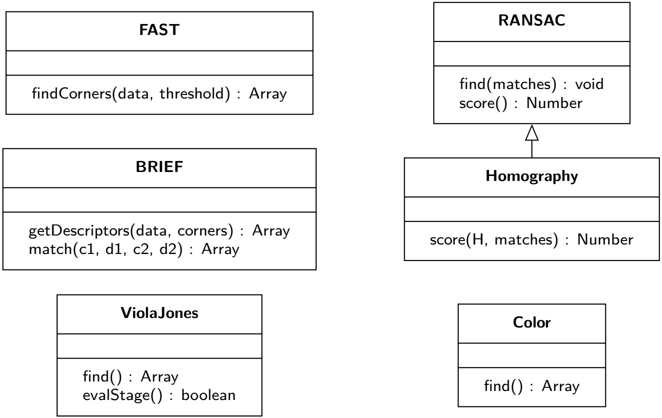
3. Markerless tracking algorithm



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

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

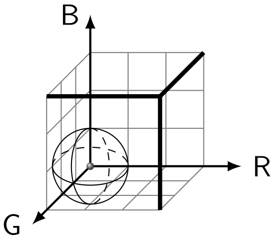
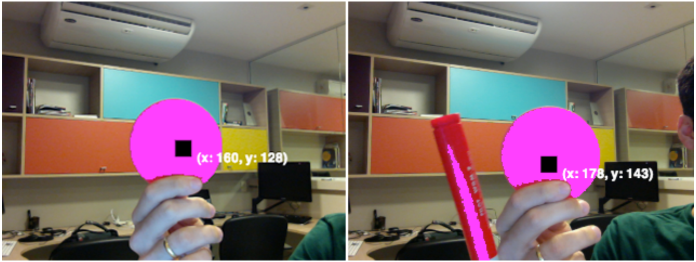
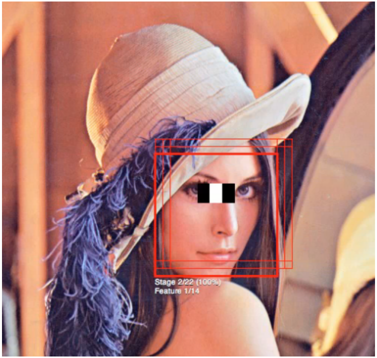


Figure : Color neighborhood represented in a RGB orthogonal three-dimensional color space.

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)

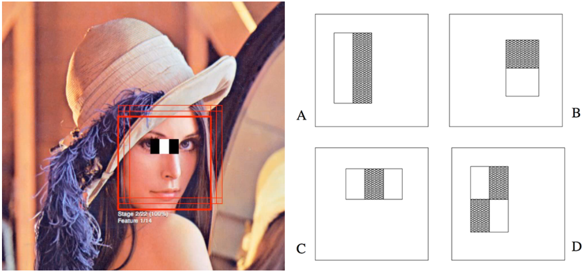


Figure : Images are classified based on the value of rectangle features.

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)

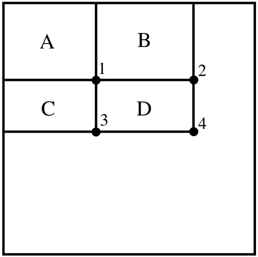


Figure : The sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$ and at location 4 is $A + B + C + D$.

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)

Optimized 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)

Optimized 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.

Minimum Neighbor Area Grouping

Simple loop trough the possible rectangles comparing the current rectangle with all other not yet compared. If their area overlaps by $\eta = 0.5$ the smallest rectangle of the set is selected.

Feature detector (FAST)

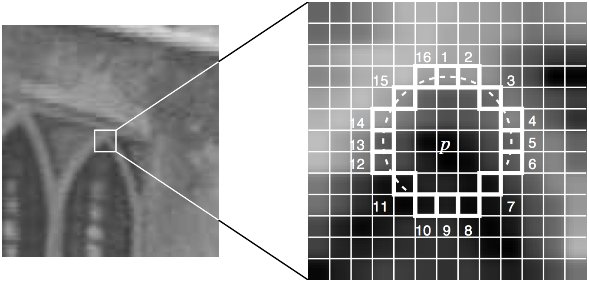
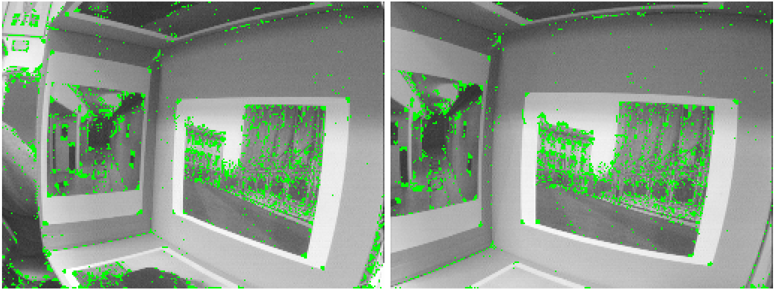
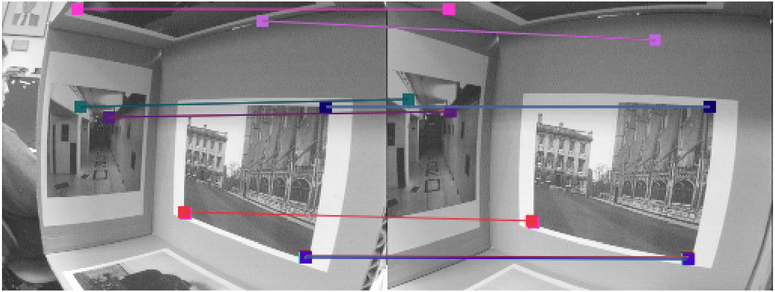


Figure : Point segment test corner detection in an image patch.

Feature detector (FAST)



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)

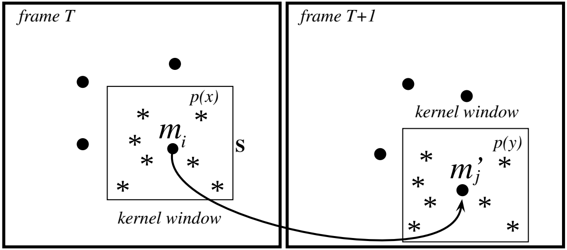


Figure : BRIEF feature extractor.

Feature extractor (BRIEF)

The weighted Hamming distance is computed by:

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

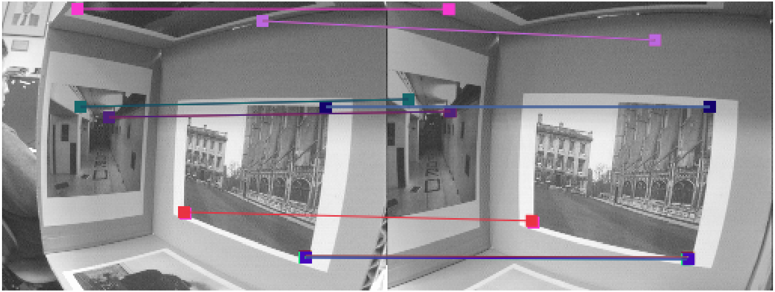
$$b_1 = 0000000001...$$

$$b_2 = 0000000011...$$

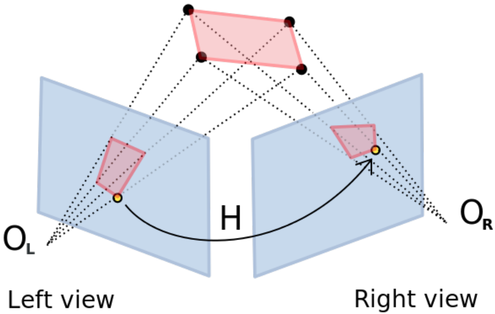
$$b_1 \otimes b_2 = 0000000010...$$

$$WHam = 1$$

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.

Homography estimation

Homography is a mapping from $P^2 \rightarrow P^2$ which is a projectivity if and only if there exists a non-singular 3×3 matrix H such that for any point in P^2 represented by vector **x** it is true that its mapped point equals $H\mathbf{x}$

$$c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad H = \begin{pmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{pmatrix},$$

where c is any non-zero constant, $(\ u\ v\ 1\ )^T$ represents **x'**, $(\ x\ y\ 1\ )^T$ represents **x**

Outline

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

Evaluation methodology

1. Examples

Technique examples.

Evaluation methodology

1. Examples

Technique examples.

2. Performance

Frames per second (FPS) metric.

Evaluation methodology

1. Examples

Technique examples.

2. Performance

Frames per second (FPS) metric.

3. Partial occlusion robustness

Examples of how the visual tracking techniques behaves under partial occlusion.

Examples

Figure : Library implementation of color tracking for different objects: On the left a red pencil marker; on the center a Rubik's magic cube from the red face; and on the right a red Ball of Whacks.

Performance

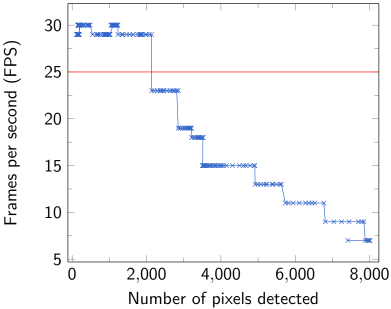


Figure : Color tracking technique tested with different numbers of pixels detected.

Oclusion robustness

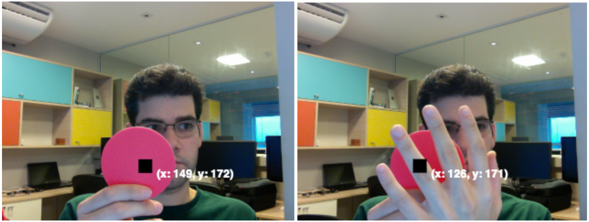


Figure : Library implementation of color tracking technique partial occlusion robustness.

Examples

Figure : Library implementation of Viola Jones using different training datas for detecting faces, eyes and mouth.

Examples

Figure : Library implementation of Viola Jones detecting multiple faces inside the real-time limit of 25 FPS.

Performance

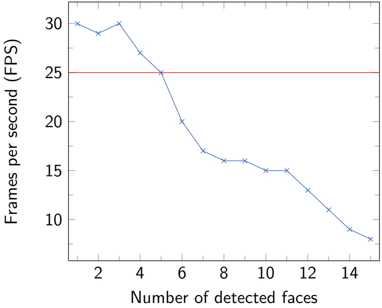


Figure : Library implementation of Viola Jones tested with different numbers of detected faces.

Oclusion robustness

Figure : Library implementation of Viola Jones partial occlusion robustness.

Examples

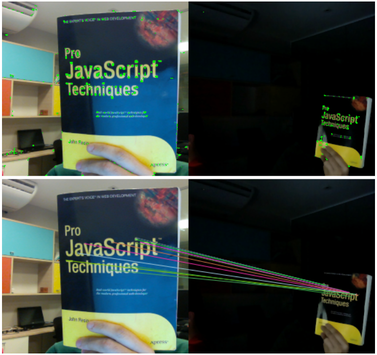


Figure : Library implementation of markerless tracking technique.

Performance

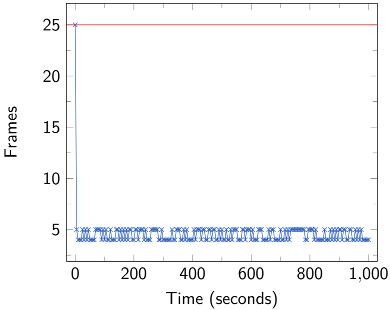


Figure : Library implementation of markerless tracking FPS metric. Maximum rate using this technique on the web is 6 FPS.

Oclusion robustness

Figure : Library implementation of markerless tracking technique partial occlusion robustness.