# 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

- Modern web browsers can natively capture the user media

# Problem definition

- Modern web browsers can natively capture the user media
- JavaScript is a language interpreted by all web browsers

# Problem definition

- Modern web browsers can natively capture the user media
- JavaScript is a language interpreted by all web browsers
- Interpreted languages have limited computational power

# Problem definition

- Modern web browsers can natively capture the user media
- JavaScript is a language interpreted by all web browsers
- Interpreted languages have limited computational power
- Capturing and processing user media are required steps for visual tracking

# 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

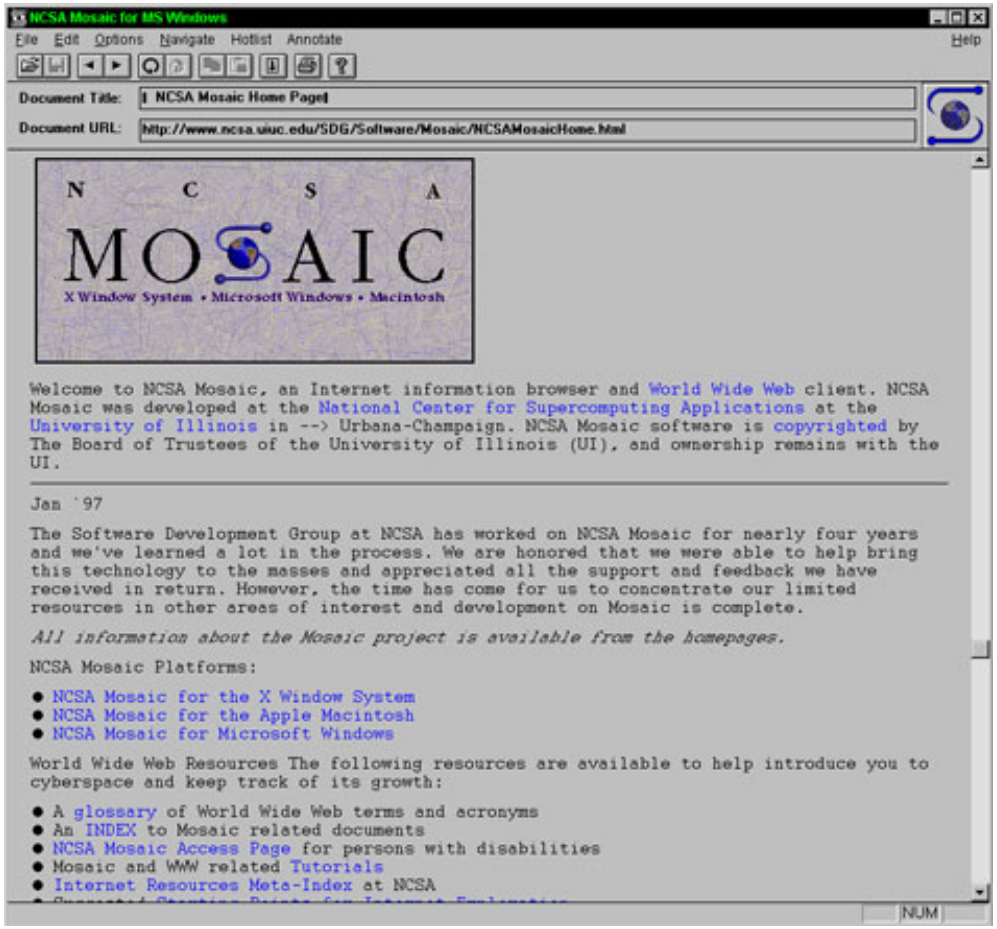
- Facilitate user interaction with the web browser
- Accelerate the use of visual tracking in commercial products
- Provide a cross-platform tracking library

# Objectives

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

# Outline

# The beggining of the web



# The beggining of the web

- Plain text and images were the most advanced features

# The beggining of the web

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

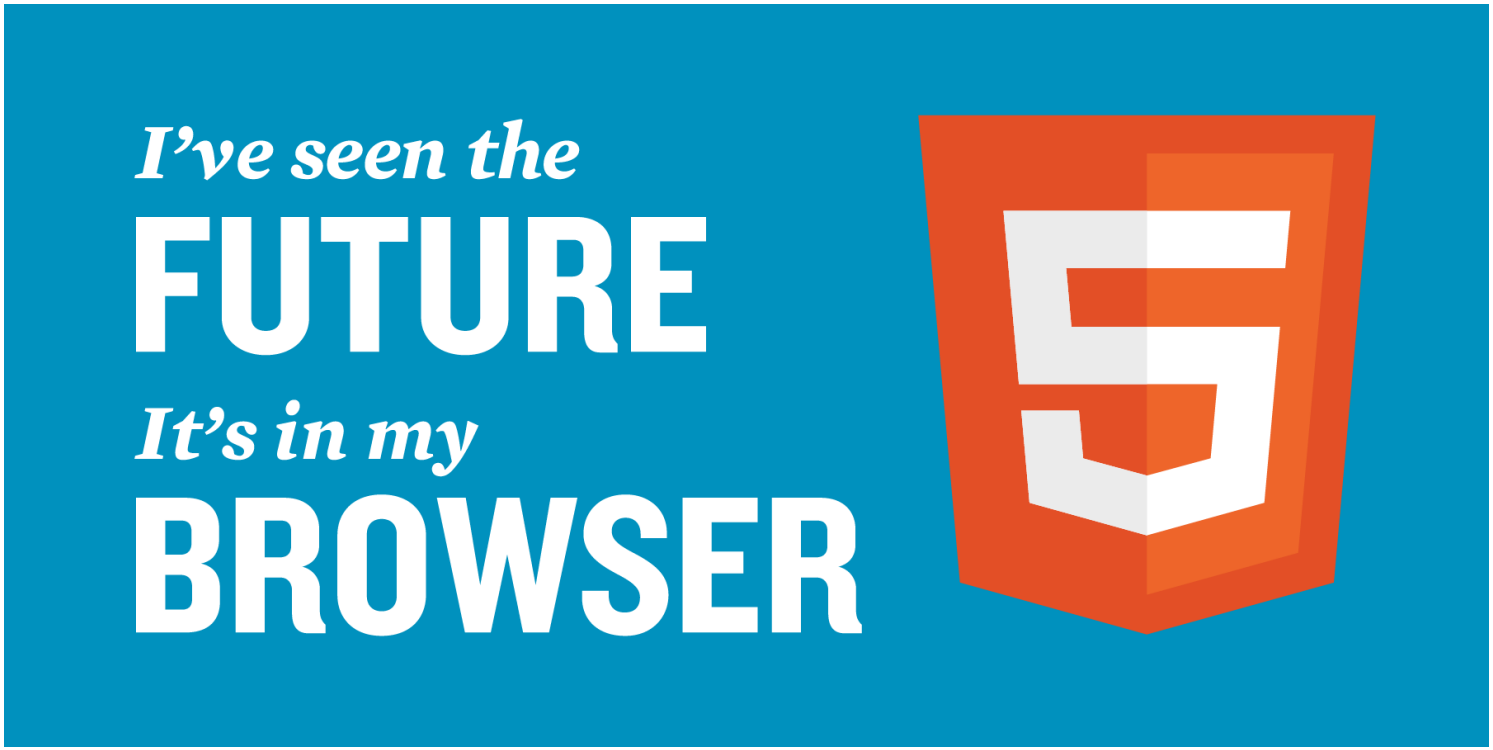
# The beggining of the web

- 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 beggining 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 technologies

- Web pages are written using HyperText Markup Language

# Browser technologies

- Web pages are written using HyperText Markup Language
- Web can be augmented with other technologies

# Browser technologies

- Web pages are written using HyperText Markup Language
- Web can be augmented with other technologies
- Layout and style information uses Cascading Style Sheets

# Browser technologies

- Web pages are written using HyperText Markup Language
- Web can be augmented with other technologies
- Layout and style information uses Cascading Style Sheets
- JavaScript is the main programming language

# Browser architecture


Figure : Web browsers running on different devices

# Browser architecture

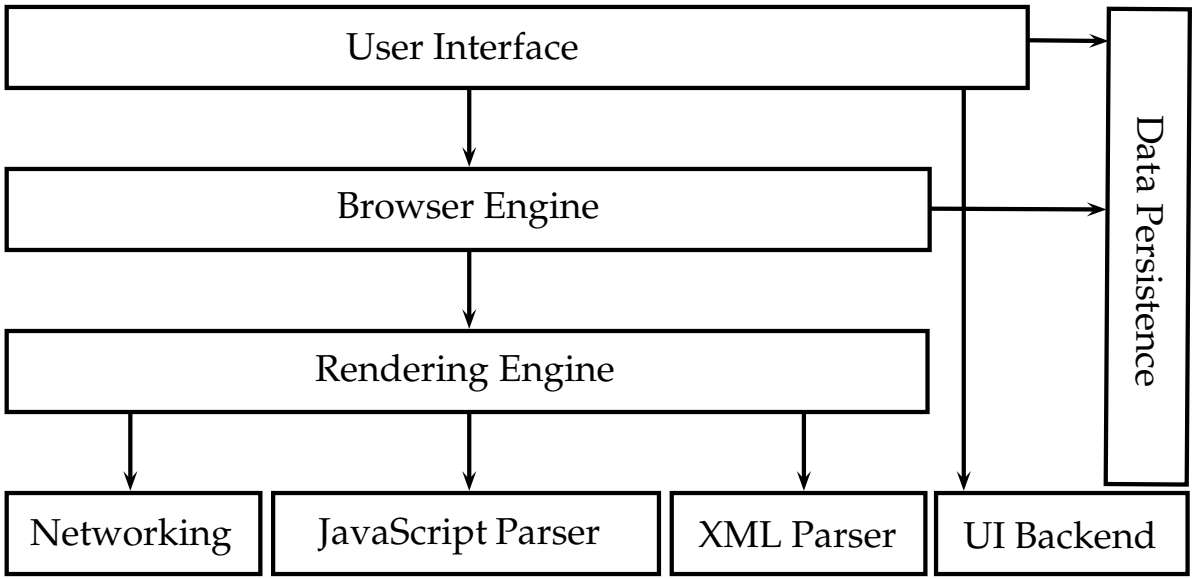



Figure : Reference architecture for web browsers

# Video capturing and processing workflow

# Audio and video


Figure : Video and audio HTML5 elements

# Capturing audio and video

# Capturing audio and video

```
1
2     <video autoplay></video>
3     <script>
4       var video = document.querySelector('video');
5       navigator.getUserMedia({video: true, audio: true}, function(localMediaStream) {
6           video.src = window.URL.createObjectURL(localMediaStream);
7           video.onloadedmetadata = function(e) { alert('Ready to go.') };
8       }, onFail);
9     </script>
```

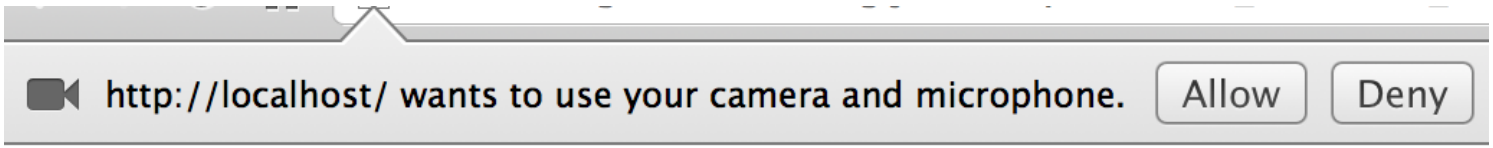
Listing 1: Capturing browser microphone and camera

# Capturing audio and video

```
1
2       <video autoplay></video>
3       <script>
4         var video = document.querySelector('video');
5         navigator.getUserMedia({video: true, audio: true}, function(localMediaStream) {
6               video.src = window.URL.createObjectURL(localMediaStream);
7               video.onloadedmetadata = function(e) { alert('Ready to go.') };
8         }, onFail);
9       </script>
```

Listing 2: Capturing browser microphone and camera

# Capturing audio and video



# Capturing audio and video



getUserMedia



<video>

[0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1]
[r g b a r g b a r g b a r g b a r g b a]





<canvas>

# Capturing audio and video

getUserMedia



<video>

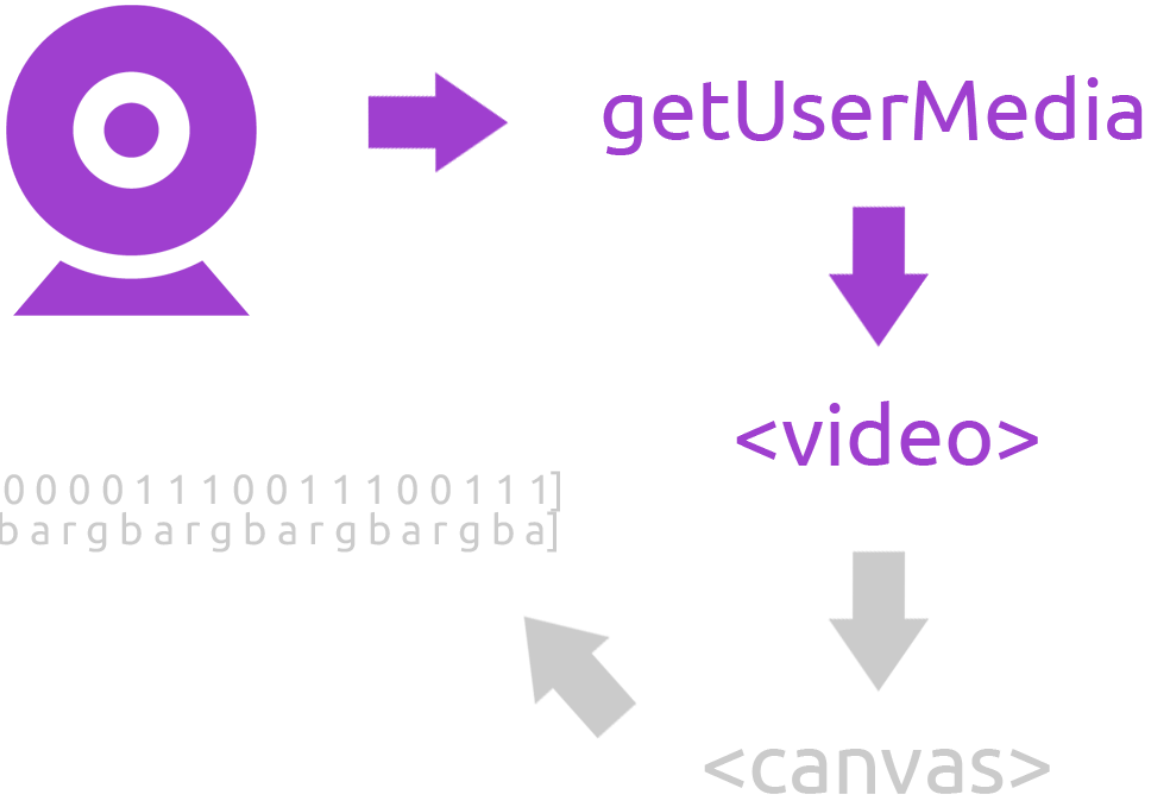
[0000011100111001111]
[rgbargbargbargbargba]



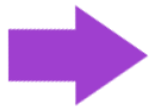


<canvas>

# Capturing audio and video



# Capturing audio and video



getUserMedia

<video>

[0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1]
[r g b a r g b a r g b a r g b a r g b a]



<canvas>

# Canvas element

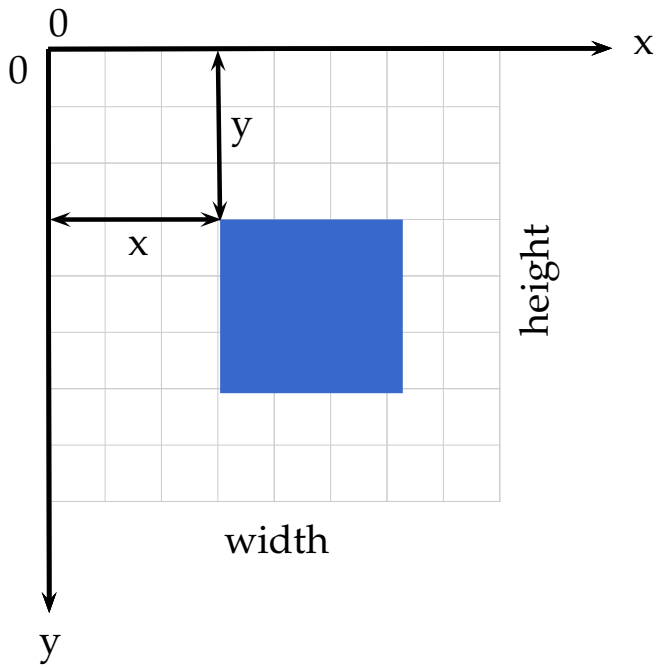



Figure : The canvas element

# Canvas element

- Introduced as a new element on HTML5

# Canvas element

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

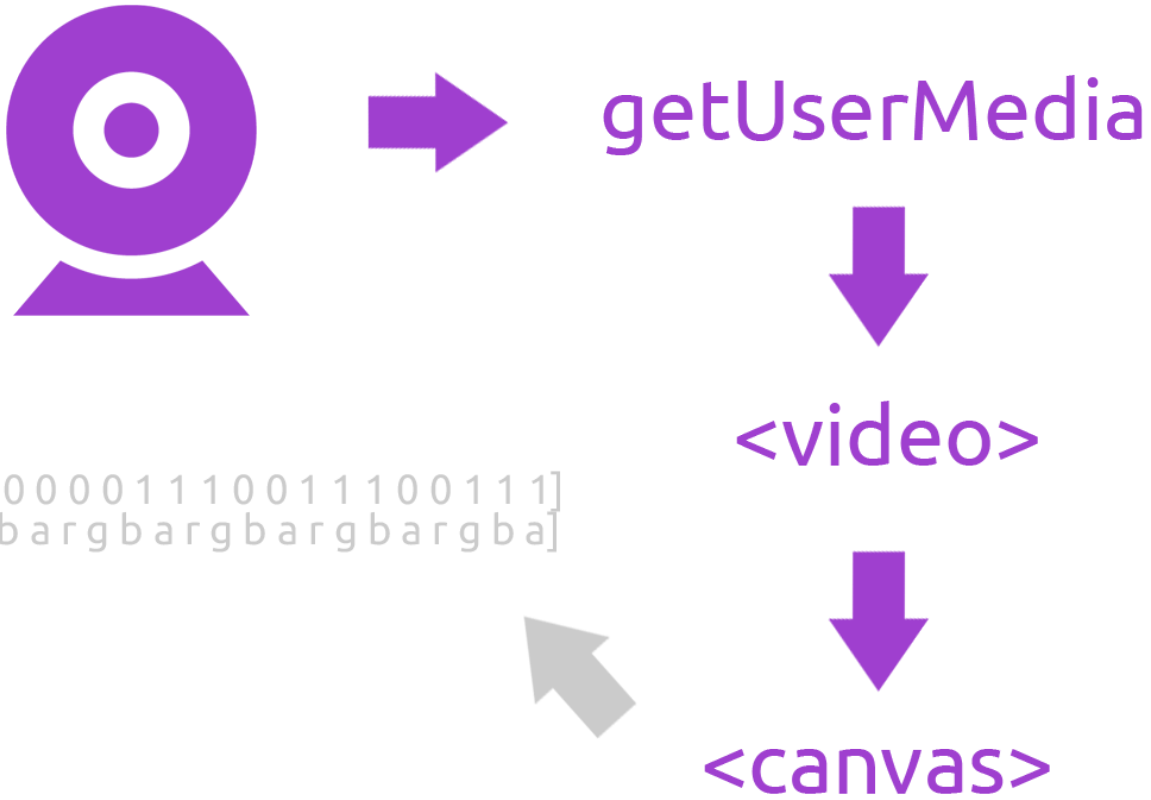
# Canvas element

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

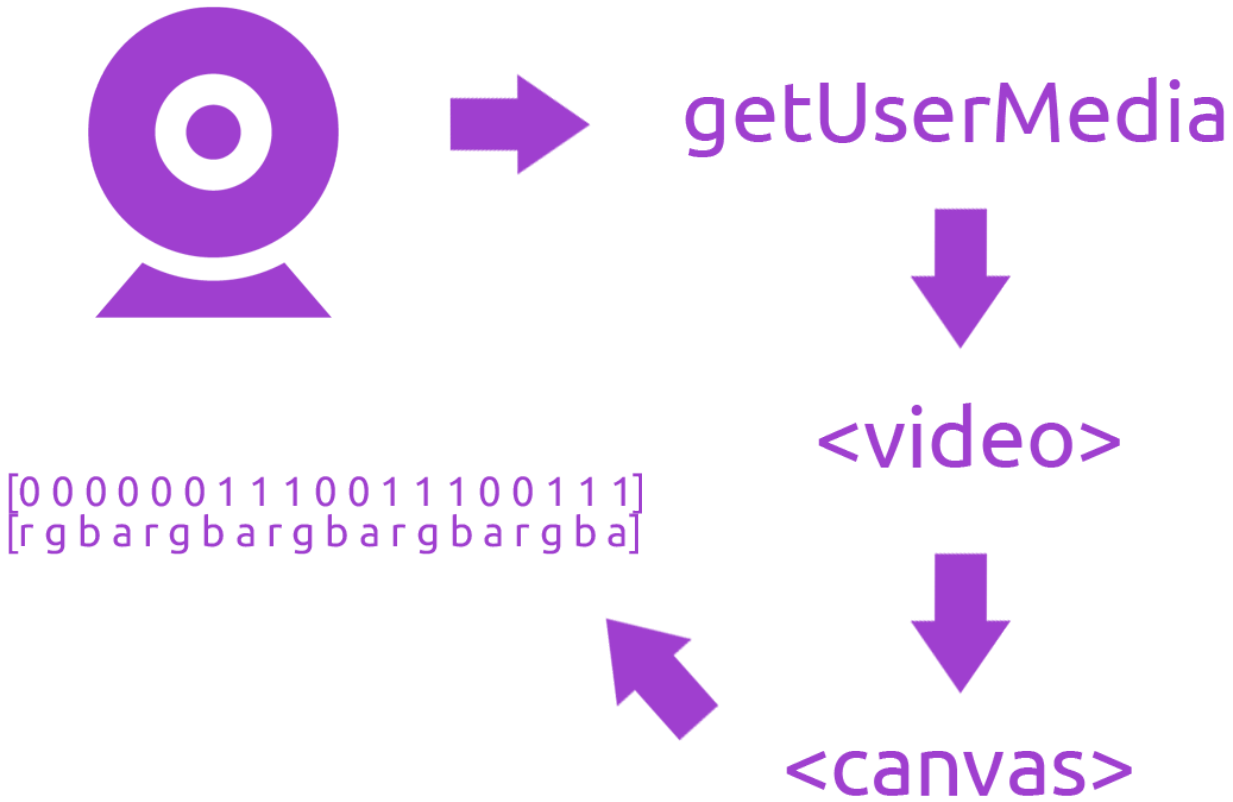
# Canvas element

- Introduced as a new element on HTML5
- Resolution-dependent bitmap canvas
- Two-dimensional grid, computer graphics coordinate system
- Can render graphs, game graphics, art, or other visual images

# JavaScript typed arrays



# JavaScript typed arrays



# JavaScript typed arrays

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

# JavaScript typed arrays

- In the past, raw data was accessed as a string
- Browsers need to quickly manipulate raw binary data

# JavaScript typed arrays

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

# JavaScript typed arrays

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

# JavaScript typed arrays

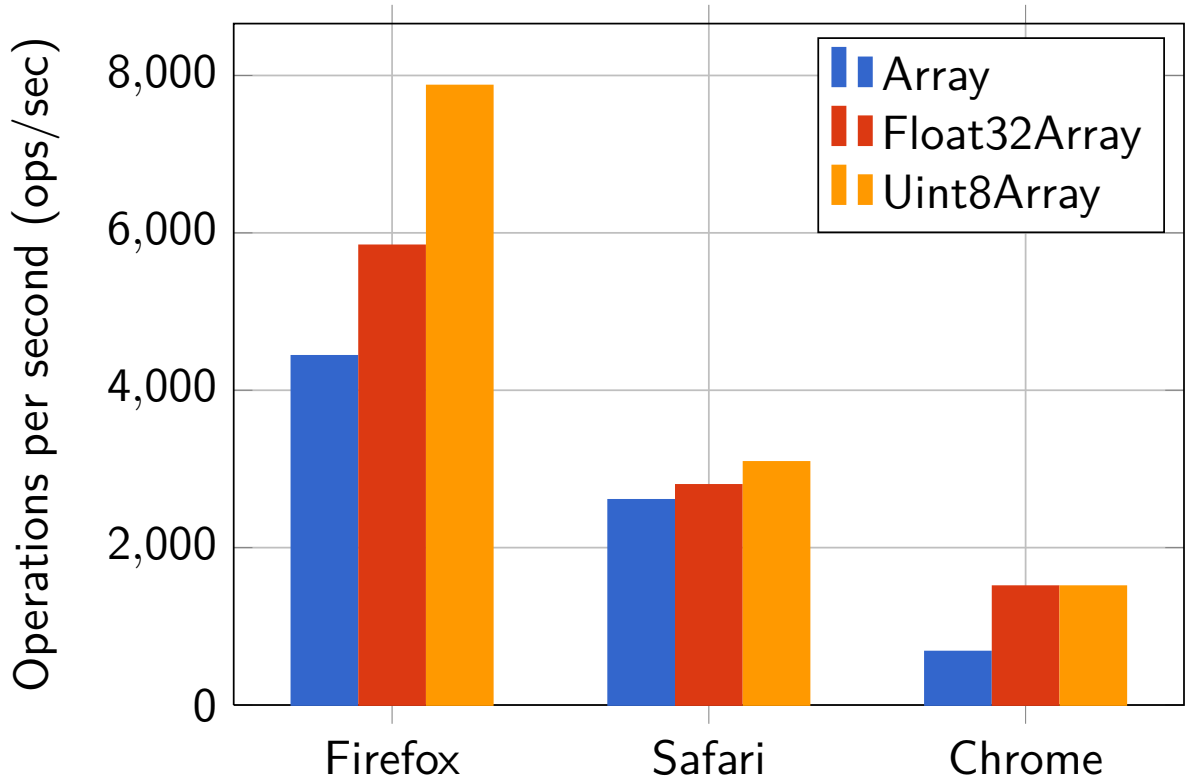



Figure : Regular *vs* typed arrays performance benchmark

# 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 $2 \times 3$ 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?

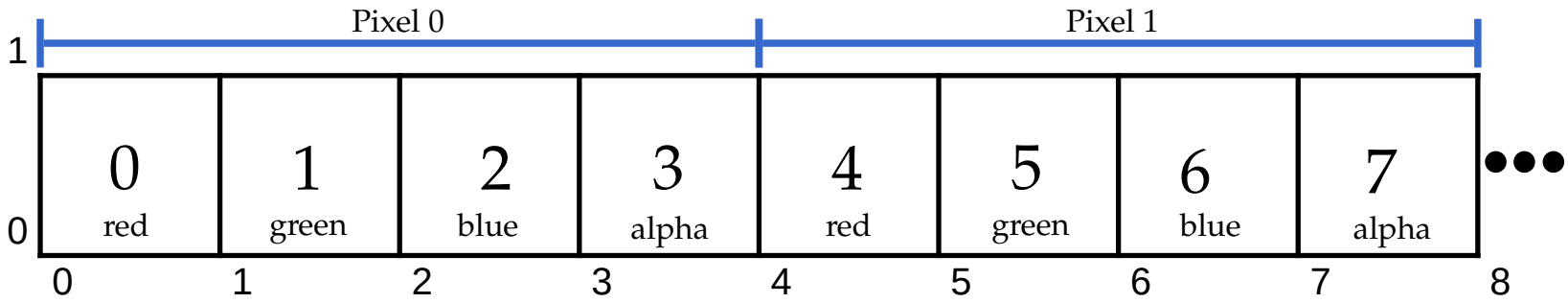
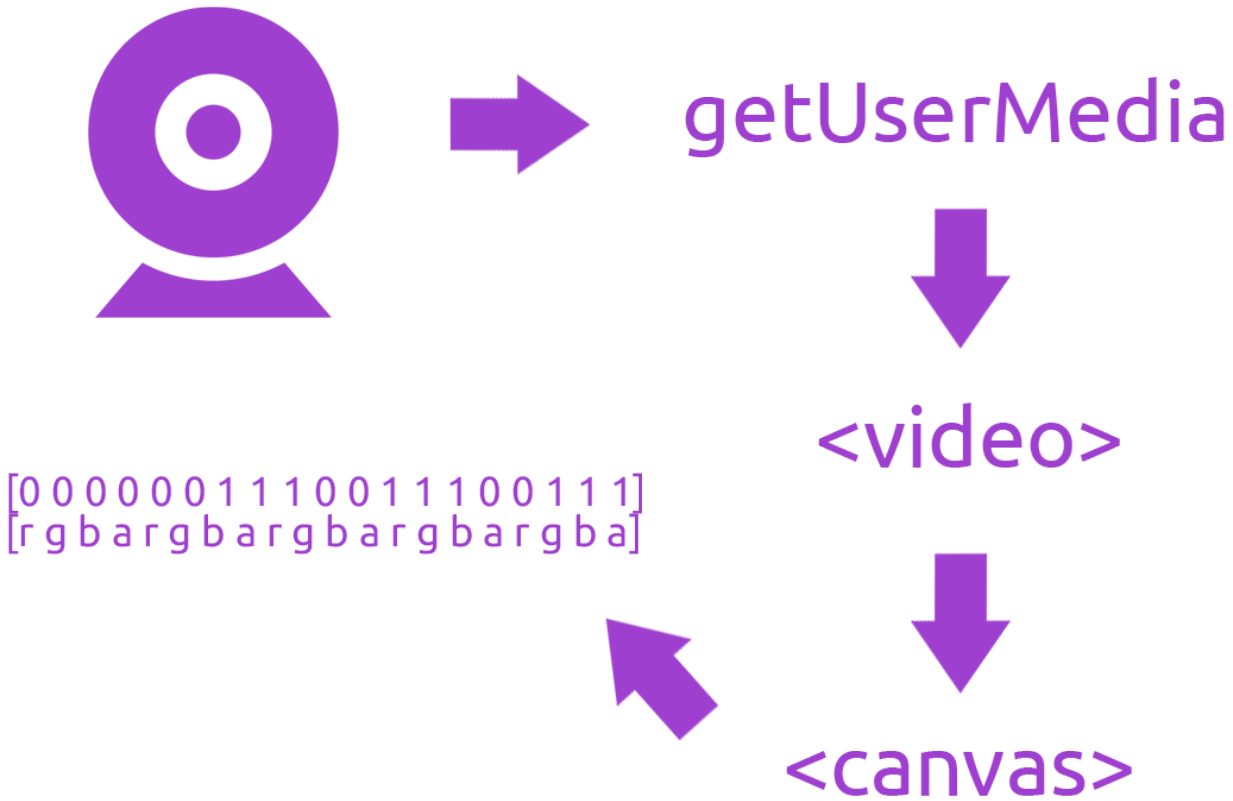



Figure : The canvas image data array of pixels

# Video capturing and processing workflow



# Video capturing and processing workflow

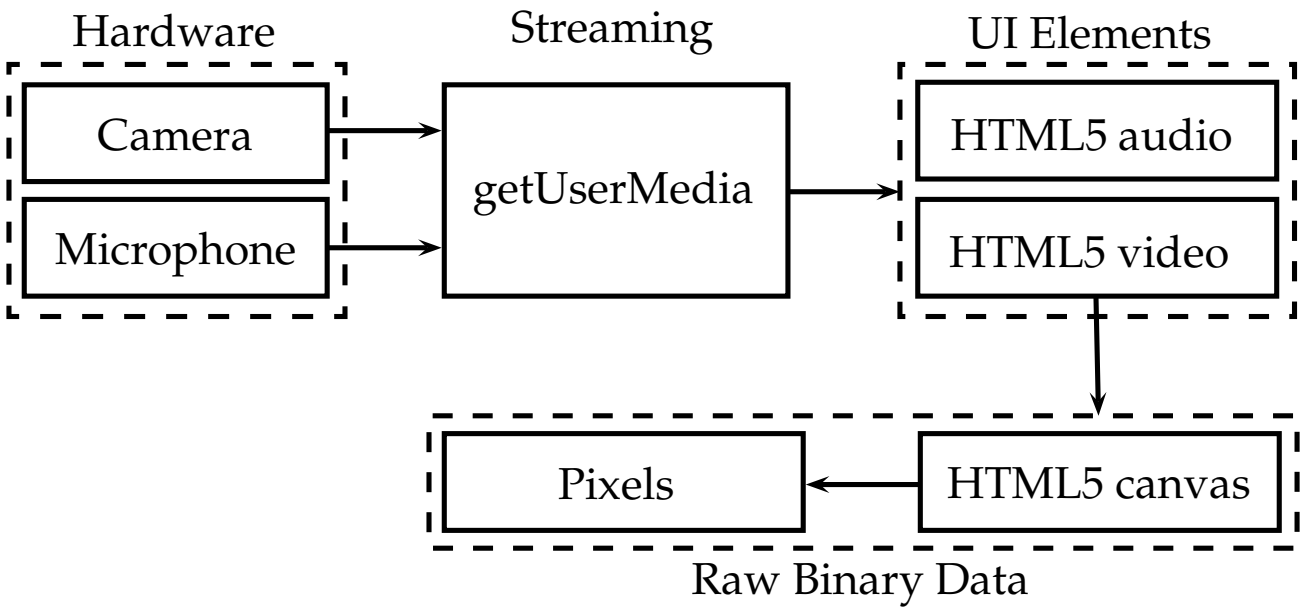



Figure : Access flow of raw binary data captured from videos

# 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

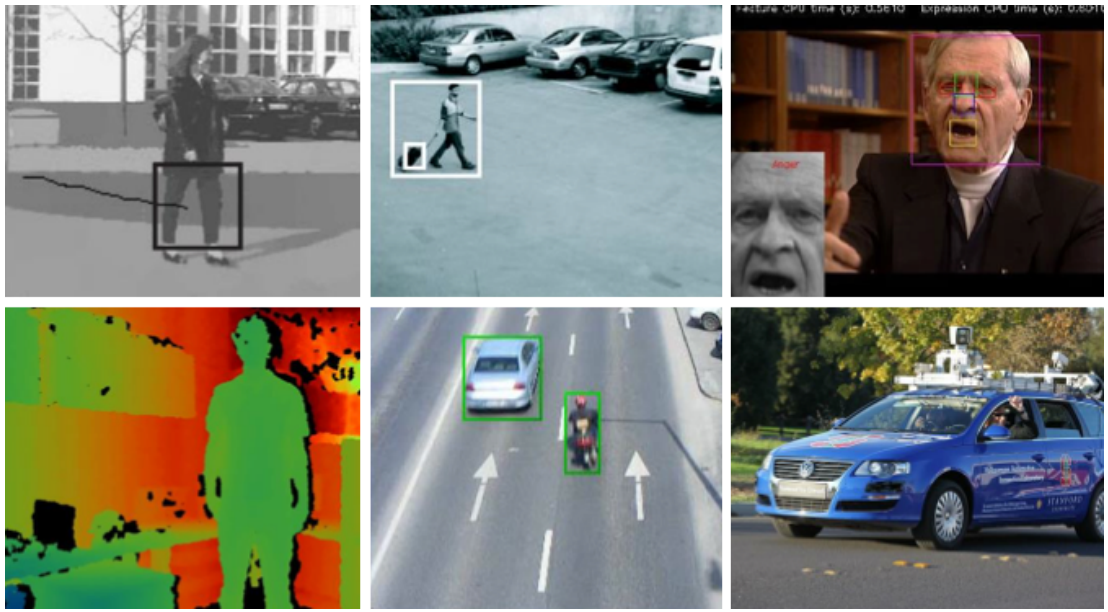
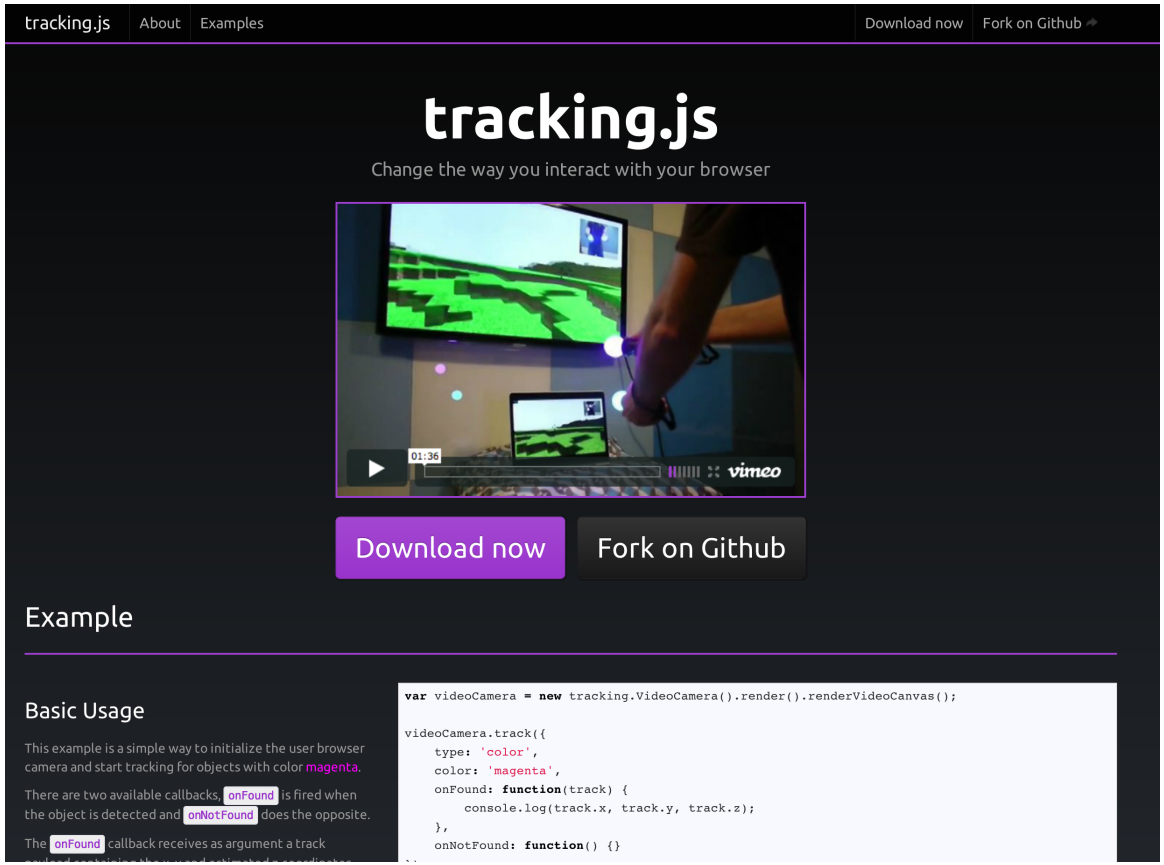


Figure : Computer vision applications: motion-based recognition (top left); automated surveillance (top center); video indexing (top right); human-computer interaction (bottom left); traffic monitoring (bottom center); vehicle navigation (bottom right).

# 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

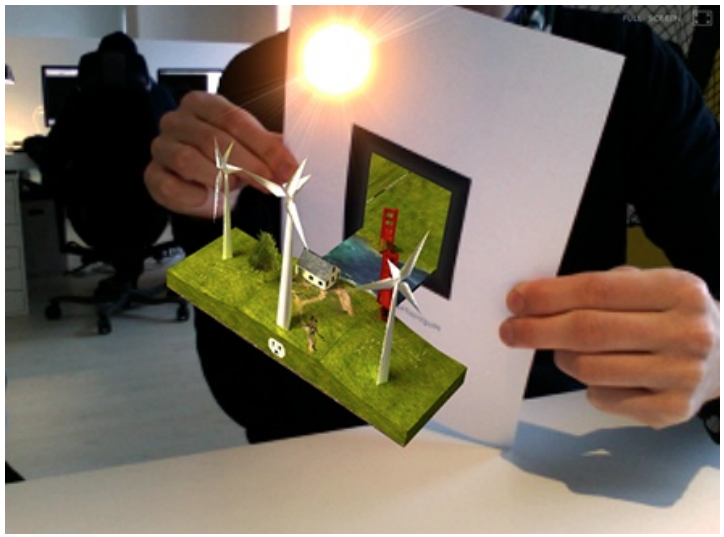


Figure : Marker based AR for the web using FLARToolKit

# Related work

- JSARToolkit: is a JavaScript port of FLARToolKit

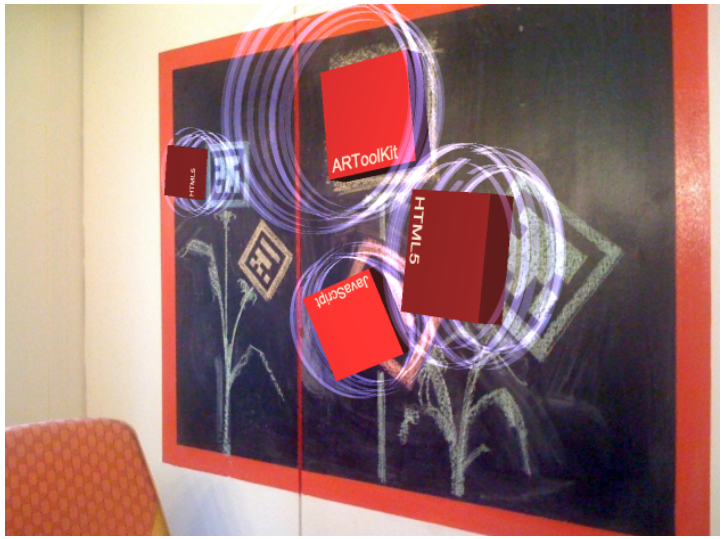



Figure : Marker-based AR for the web using JSARToolKit

# 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

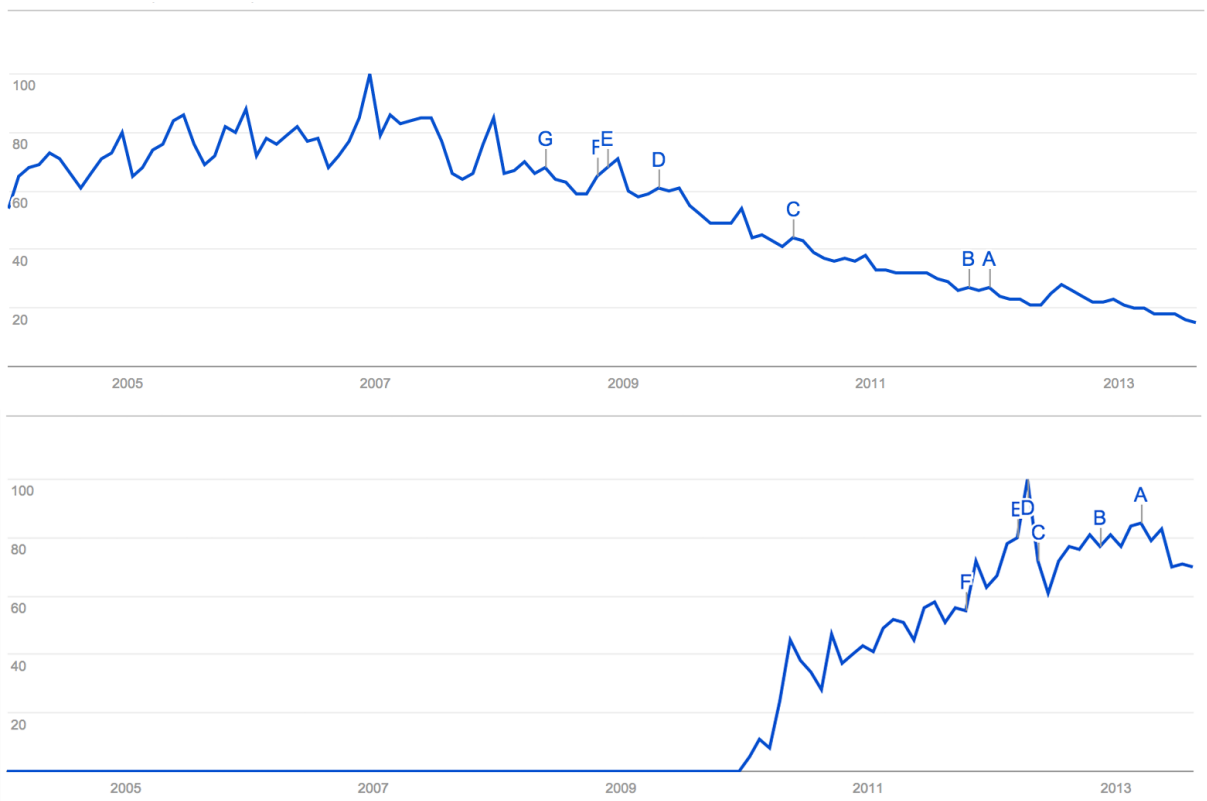



Figure : Google trends results for “flash games” (top) and for “html5 games” (bottom)

# Library features

# Library modules

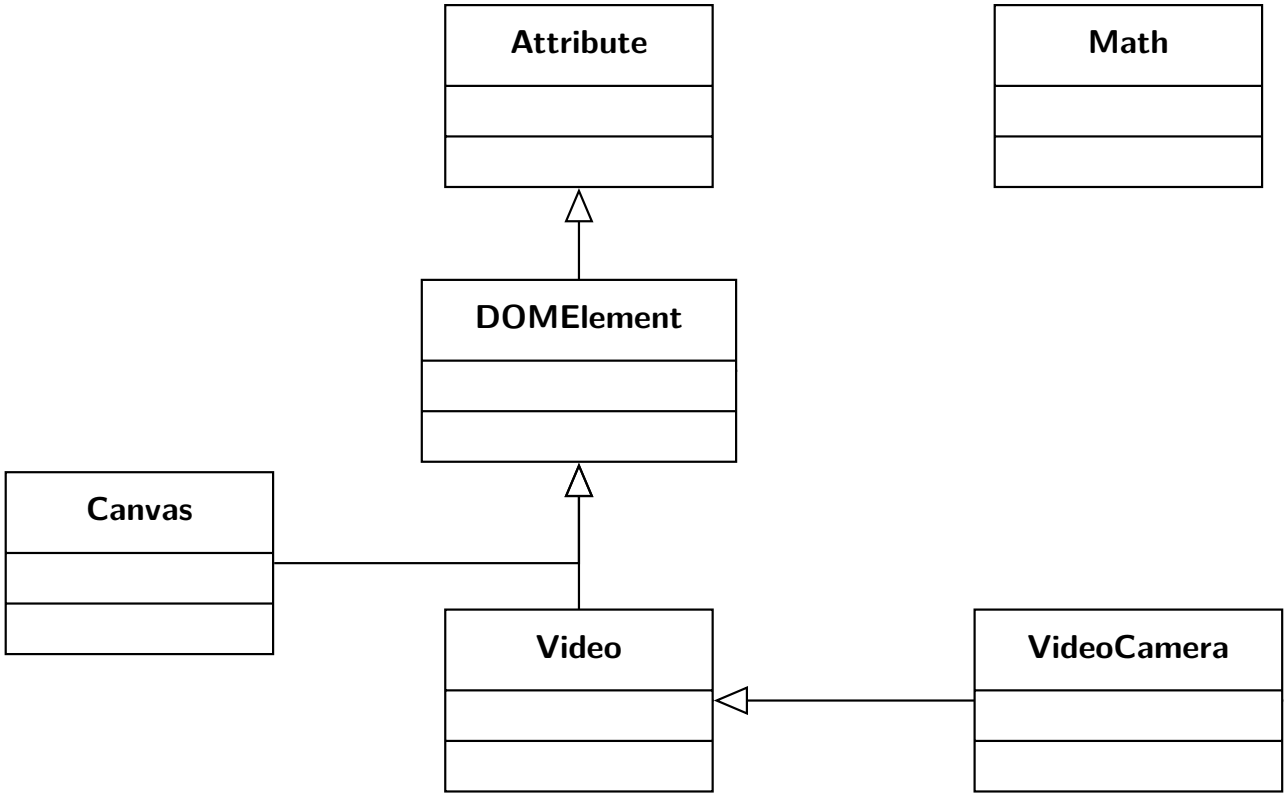



Figure : Base classes of tracking.js library

# Library modules

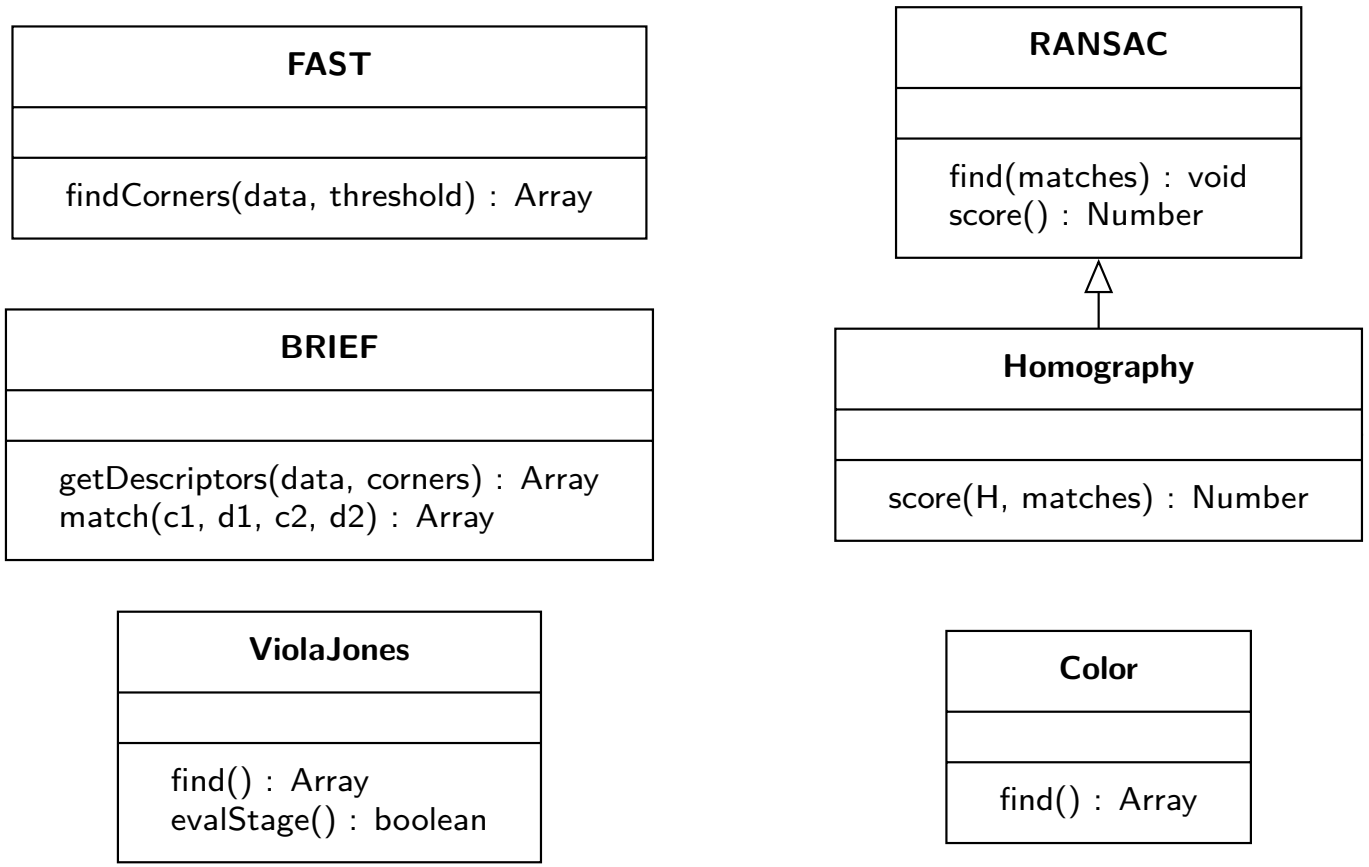
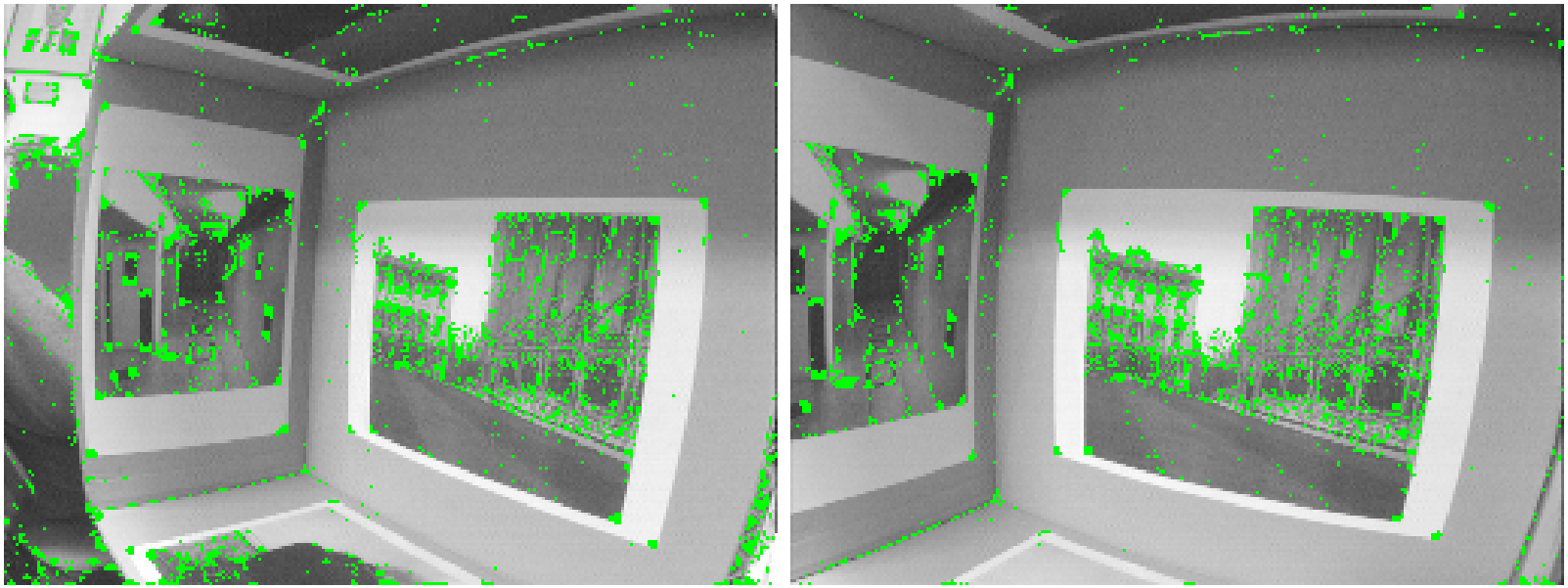



Figure : Visual tracking classes of tracking.js library

# Feature detector (FAST)



# Feature detector (FAST)

- Detects individual features across images

# Feature detector (FAST)

- Detects individual features across images
- Robustness against partial occlusions or matching errors

# Feature detector (FAST)

- Detects individual features across images
- Robustness against partial occlusions or matching errors
- Used as the first step of many vision tasks such as tracking

# Feature detector (FAST)

- Detects individual features across images
- Robustness against partial occlusions or matching errors
- Used as the first step of many vision tasks such as tracking
- Features from Accelerated Segment Test (FAST)

# Feature detector (FAST)

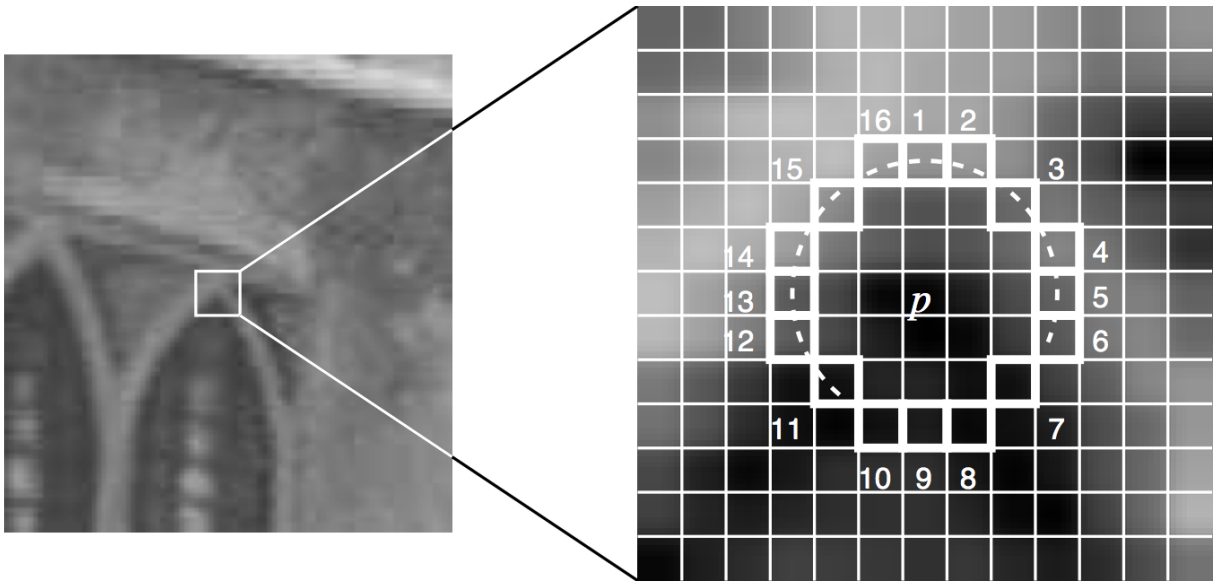
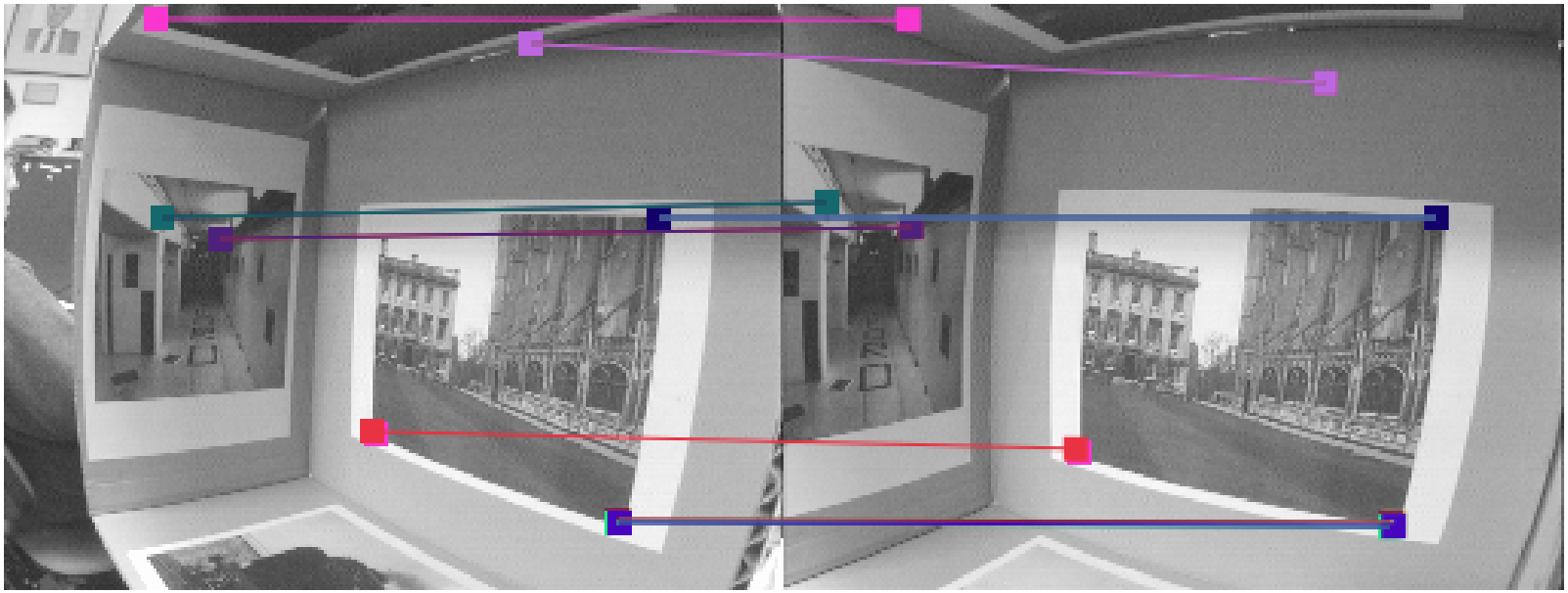



Figure : Point segment test corner detection in an image patch

# Feature extractor (BRIEF)



# Feature extractor (BRIEF)

- To estimate motion, match sets of features is required

# Feature extractor (BRIEF)

- To estimate motion, match sets of features is required
- Use binary strings to describe the keypoints

# Feature extractor (BRIEF)

- To estimate motion, match sets of features is required
- Use binary strings to describe the keypoints
- Fast to compute, to match and is memory efficient

# Feature extractor (BRIEF)

- To estimate motion, match sets of features is required
- Use binary strings to describe the keypoints
- Fast to compute, to match and is memory efficient
- Binary Robust Independent Elementary Features (BRIEF)

# Feature extractor (BRIEF)

On BRIEF to generate the binary strings it is defined the test $\tau$ 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, since it presented good matching results and performance. The number of bytes required to store the descriptor can be calculated by $k = n_d/8$.

# Feature extractor (BRIEF)

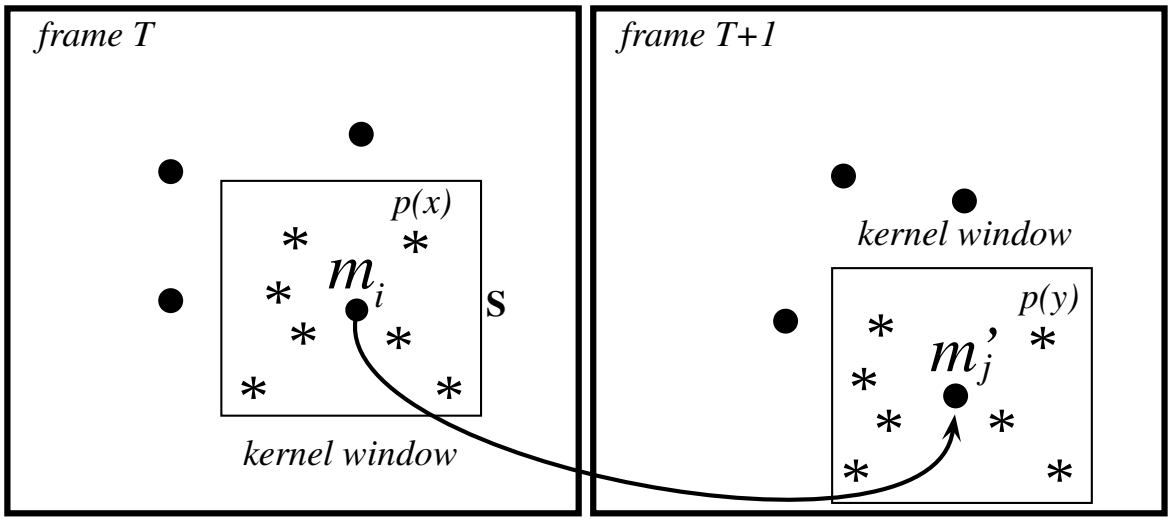



Figure : BRIEF feature extractor

# Feature extractor (BRIEF)

The weighted Hamming distance is computed by:

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

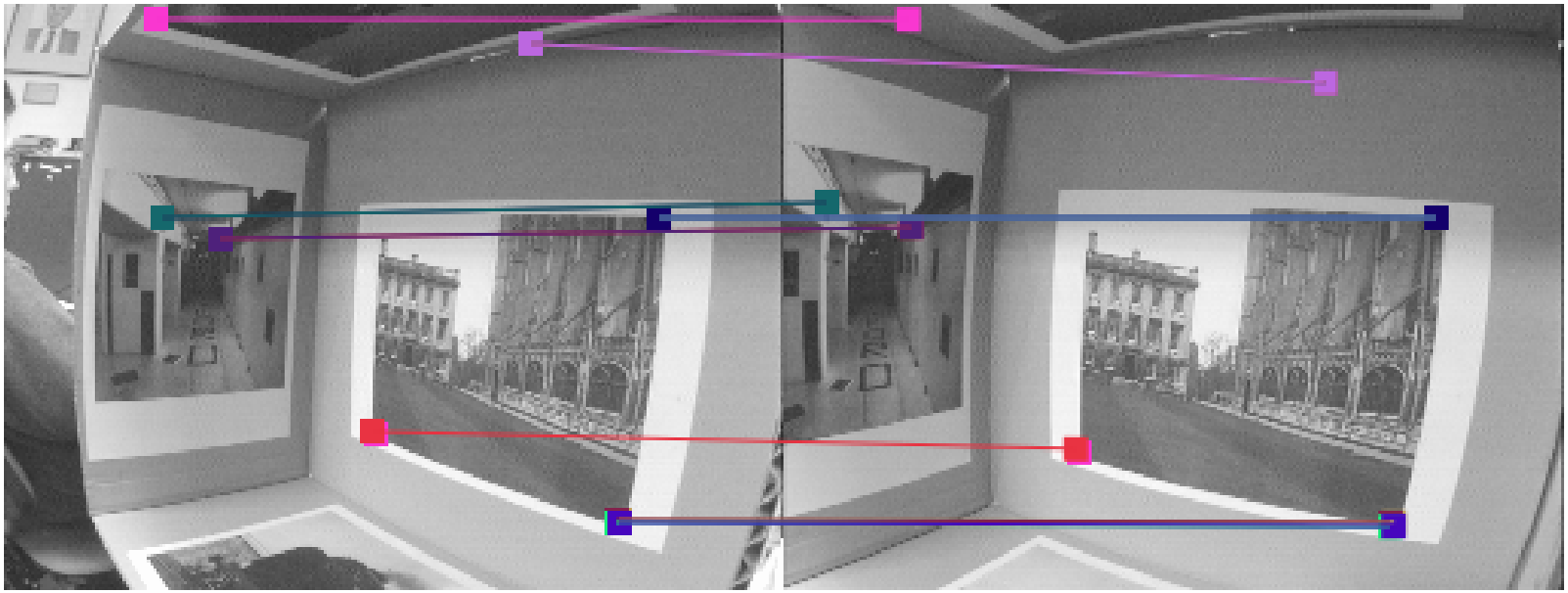
$$b_1 = 0000000001...$$

$$b_2 = 0000000011...$$

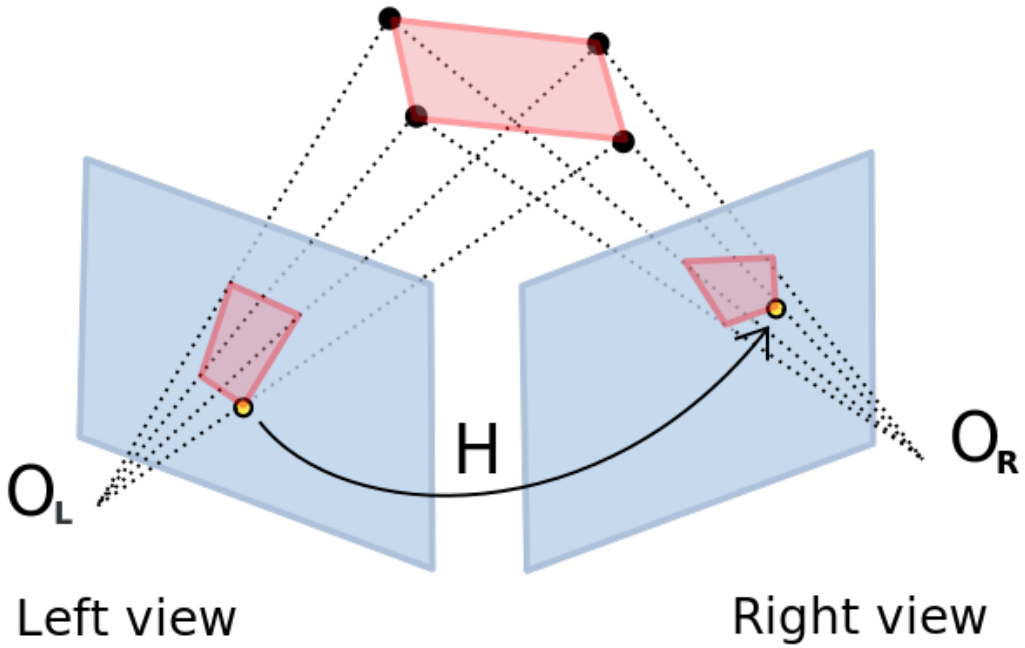
$$b_1 \otimes b_2 = 0000000010...$$

$$WHam = 1$$

# Feature extractor (BRIEF)



# Homography estimation



# 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 $3 \times 3$ matrix $H$ such that for any point in $P^2$ represented by vector $\mathbf{x}$ it is true that its mapped point equals $H\mathbf{x}$

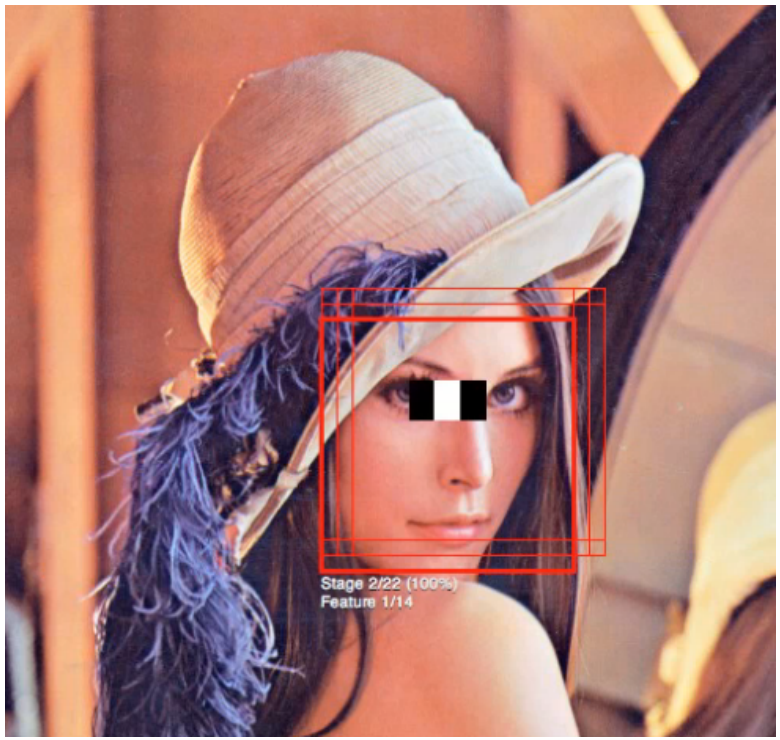
$$c \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad H = \begin{pmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{pmatrix},$$

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

# 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.

# Rapid object detection (Viola Jones)



# 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)

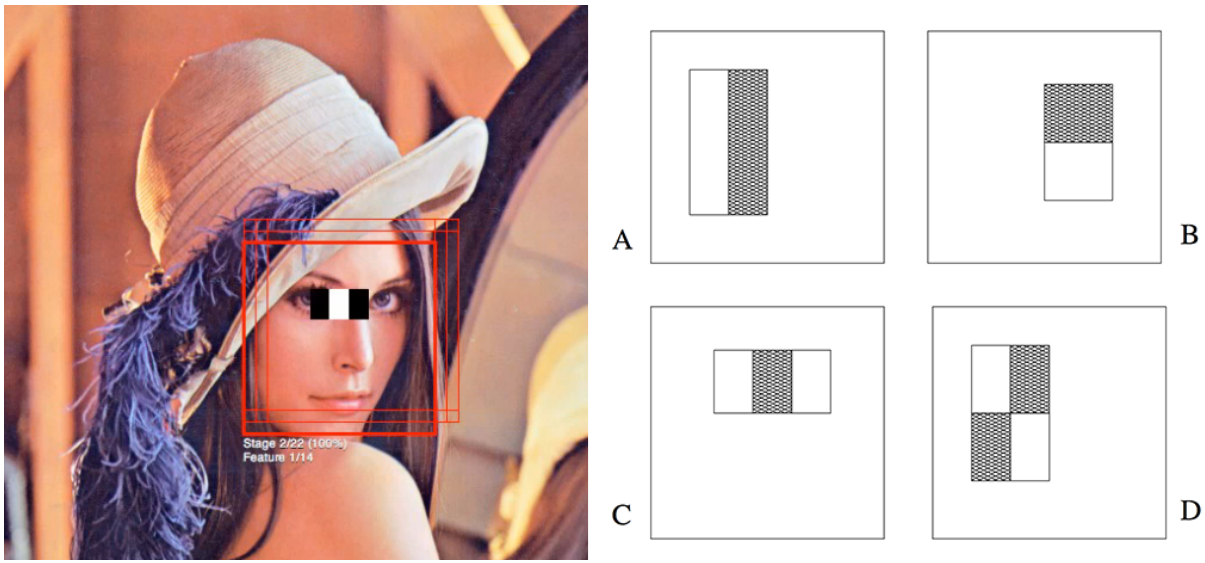



Figure : Images are classified based on the value of rectangle features

# 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)

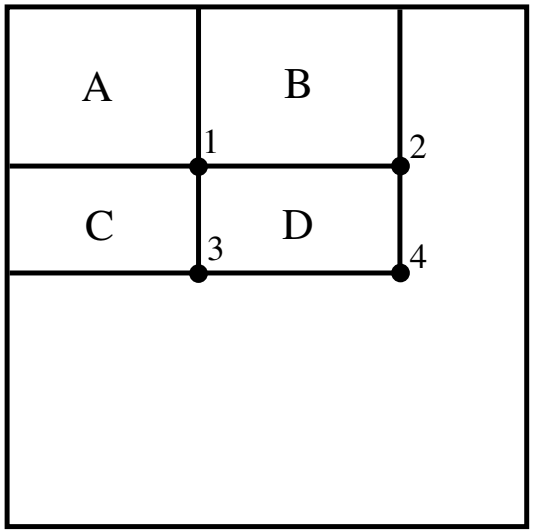



Figure : The sum of the pixels within rectangle $D$ can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle $A$. The value at location 2 is $A + B$, at location 3 is $A + C$ and at location 4 is $A + B + C + D$

# Rapid object detection (Viola Jones)

## Scanning detector algorithm

# Rapid object detection (Viola Jones)

## Scanning detector algorithm

1. Create or scale a $20 \times 20$ squared block by 1.25 per iteration

# Rapid object detection (Viola Jones)

## Scanning detector algorithm

1. Create or scale a $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ pixels over the image

# Rapid object detection (Viola Jones)

## Scanning detector algorithm

1. Create or scale a $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ 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 $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ 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 $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ 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 $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ 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 $20 \times 20$ squared block by 1.25 per iteration
2. Loop the block by $\Delta$ 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.