LABORATORIO DI REALTÀ AUMENTATA

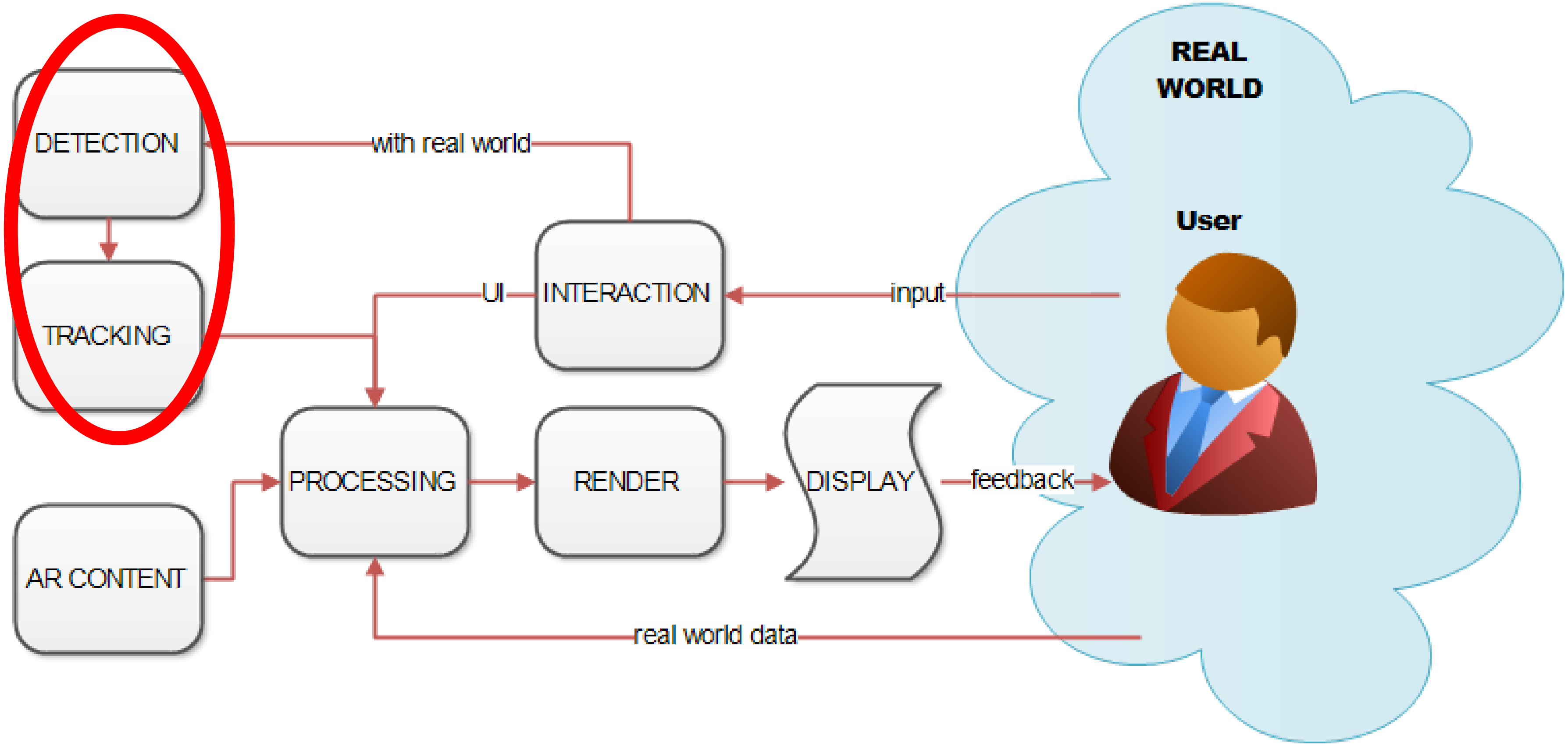
Claudio Piciarelli

Università degli Studi di Udine

Corso di Laurea in Scienze e Tecnologie Multimediali

Project: marker detection

Architecture of an AR system



STM AR: marker detection

- We already know how to acquire images from a webcam
- We will now process these images to detect fiducial markers

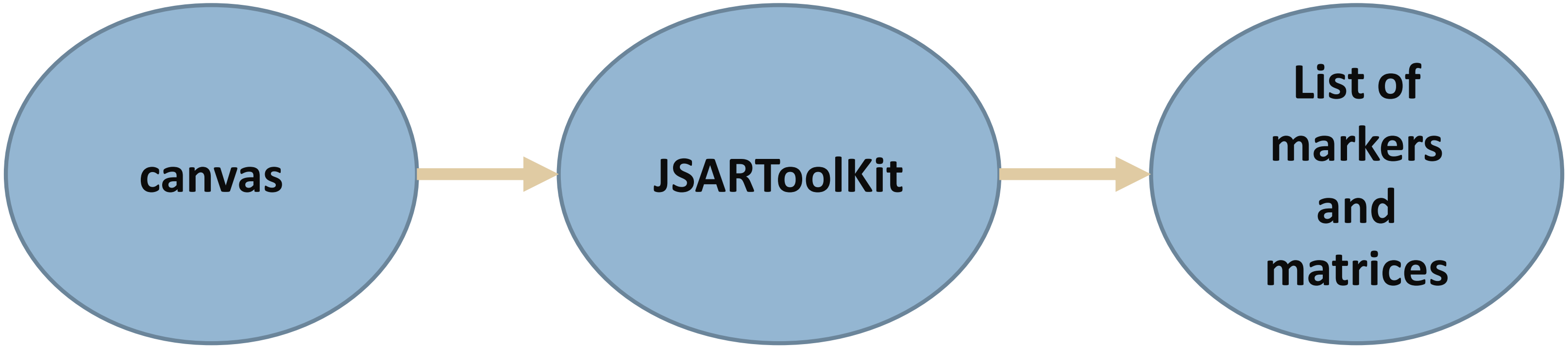
- library used: **JSARToolKit**

JSARToolKit

- Open source library for fiducial marker detection and tracking
- Port of the Flash FLARToolkit library…
- Which was a port of the Java NyARToolkit lib…
- Which was a port of the C ARToolKit lib

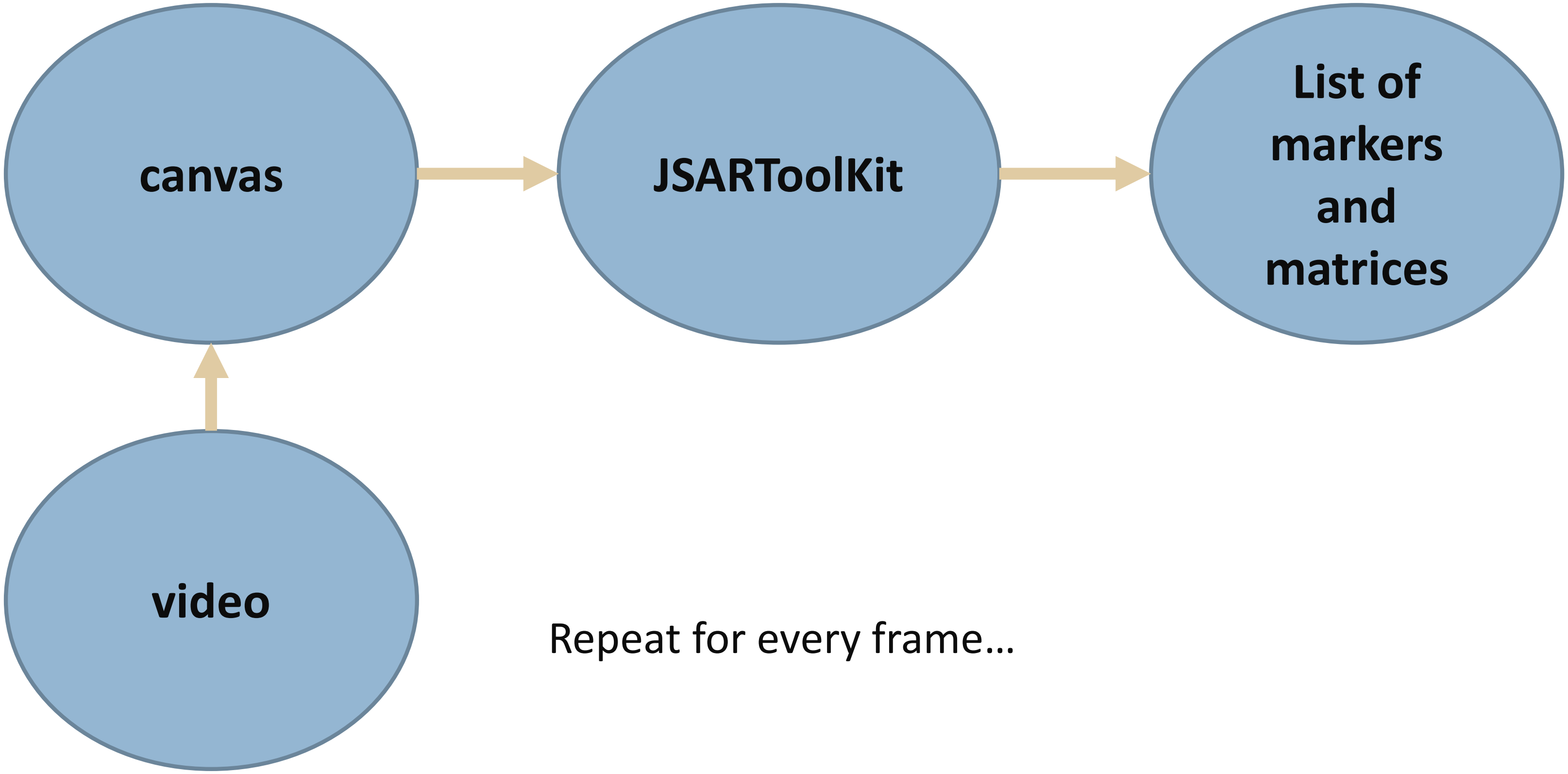
JSARToolKit

- How it works
  - Detection on a single image



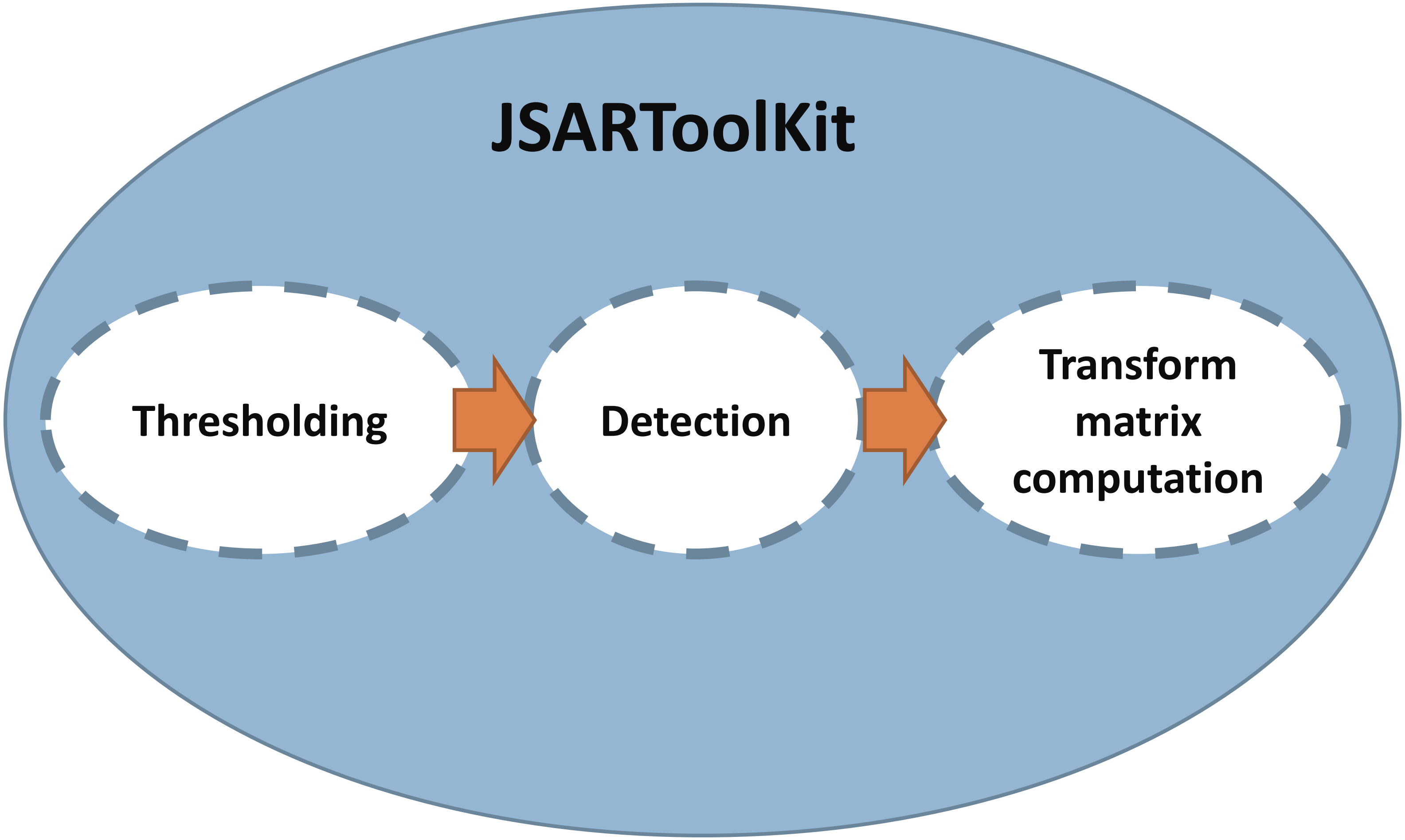
JSARToolKit

- How it works
  - Detection on a video

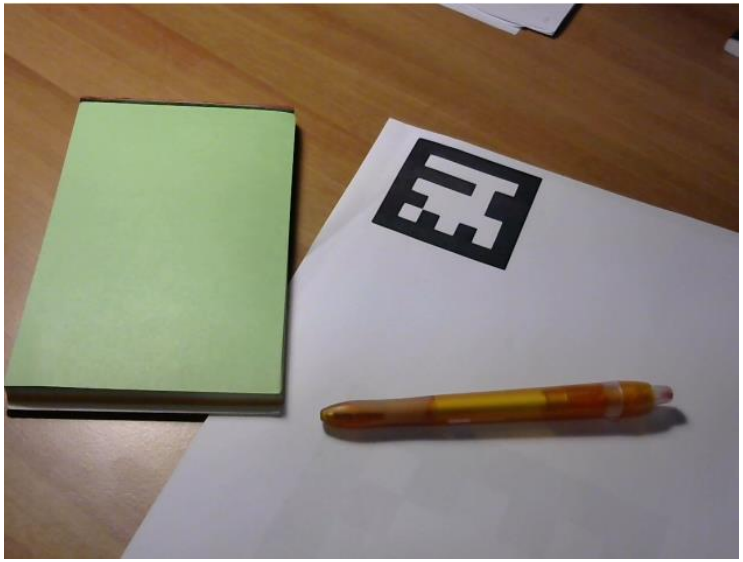


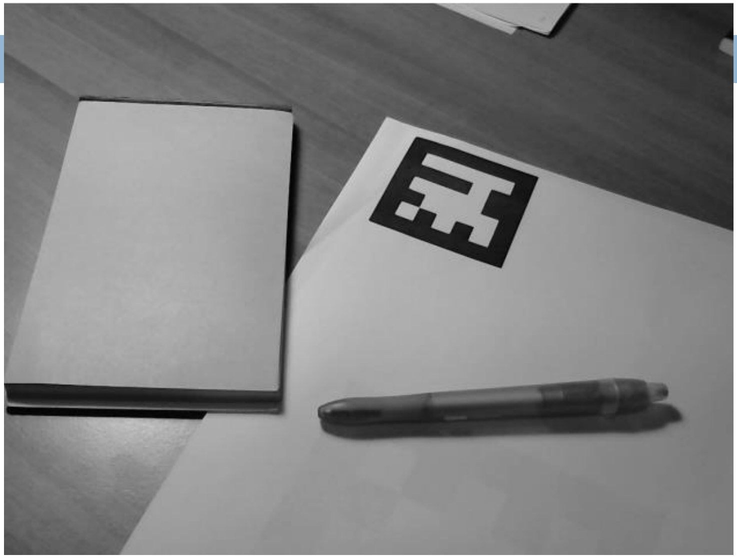
Repeat for every frame…

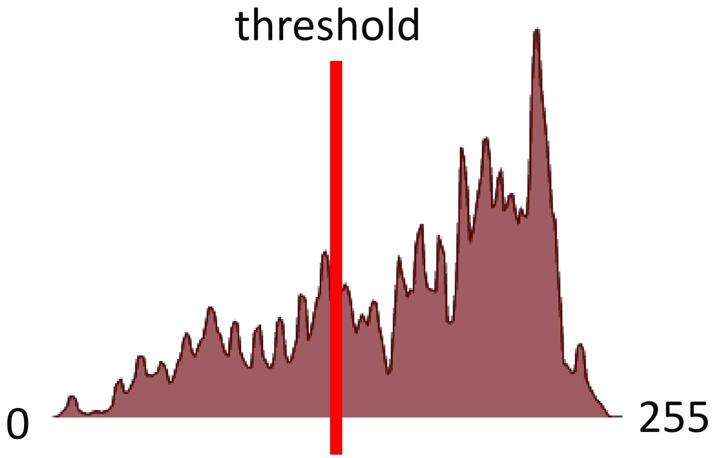
JSARToolKit internals

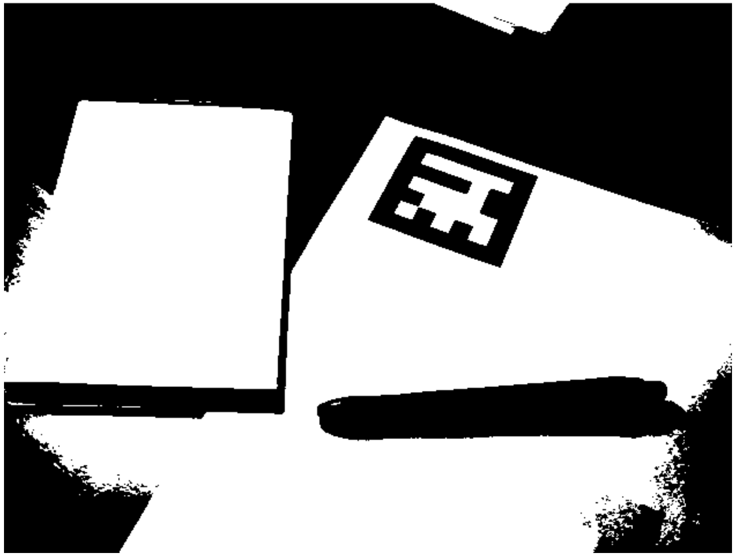


Thresholding

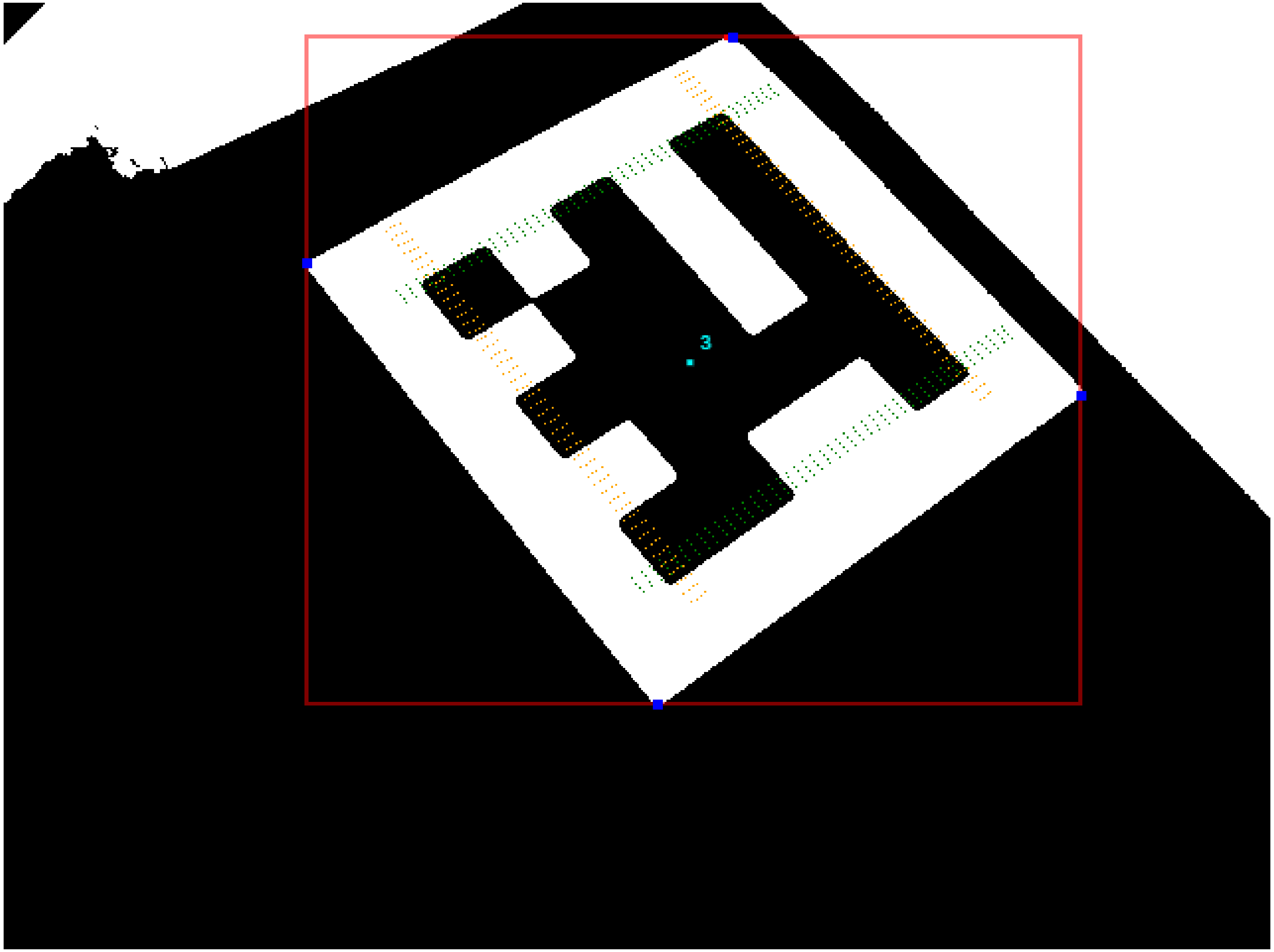




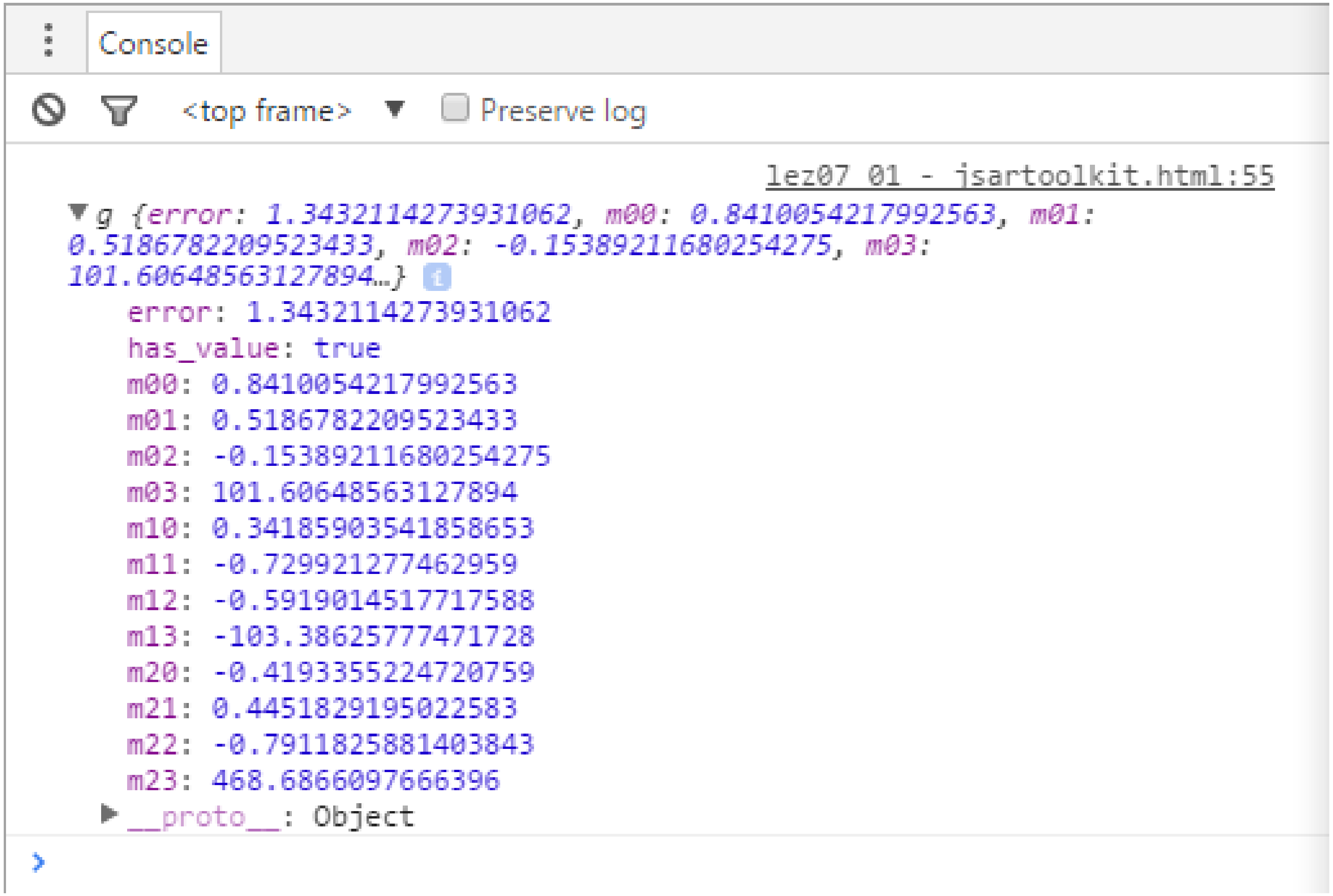




Detection



Matrix computation



```
                                                   lez07 01 - jsartoolkit.html:55
▼g {error: 1.3432114273931062, m00: 0.8410054217992563, m01:
0.5186782209523433, m02: -0.15389211680254275, m03:
101.60648563127894…}
    error: 1.3432114273931062
    has_value: true
    m00: 0.8410054217992563
    m01: 0.5186782209523433
    m02: -0.15389211680254275
    m03: 101.60648563127894
    m10: 0.34185903541858653
    m11: -0.729921277462959
    m12: -0.5919014517717588
    m13: -103.38625777471728
    m20: -0.4193355224720759
    m21: 0.4451829195022583
    m22: -0.7911825881403843
    m23: 468.6866097666396
  ▶ __proto__: Object
›
```

Getting JSARToolKit

- Download it from the elearning page
- …or download it from:
  - **`https://github.com/kig/JSARToolKit`**
- Use the stripped down version:
  - JSARToolKit.min.js

Markers

- JSARToolKit supports its own fiducial markers, which encode numbers
- You can find markers for the numbers 0 to 99 in the JSARToolKit zip you just downloaded
  - Folder: demos/markers
- These are .png files (rename them if you can’t open)
- Print them, or keep one on screen, or use the marker.webm video file…

Including JSARToolKit in our project

- Open the project of the previous lesson
- Import the javascript code with:

```
<script src="JSARToolKit.min.js"></script>
```

- Copy the JSARToolKit.min.js file in the same folder of your html page!

Adding a canvas

- As mentioned before, JSARToolKit works on canvas, not videos
- Add a canvas to the HTML page:

```
<body>
       <video autoplay id="myvideo"></video>
       <canvas id="mycanvas"></canvas>
</body>
```

Wait for the video to be ready

- We must wait the video is loaded (or at least its *metadata*) before we can start processing it
- Metadata contains duration, dimensions, subtitles tracks etc…
- We listen for the **`loadedmetadata`** event
- For a full list of video events, see:

  `http://www.w3schools.com/tags/ref_av_dom.asp`

Event listener

- After the call to **`getUserMedia`**, set the listener

```
    window.onload = function(){
            // connect to webcam
            var video = document.getElementById("myvideo");
            var constraints = {audio: false, video: true};
            navigator.mediaDevices.getUserMedia(constraints)
            .then(function(stream){
                    video.srcObject = stream;
            })
            .catch(function(err){
                    alert(err.name + ": " + err.message);
                    video.src = "marker.webm";
            });
            video.onloadedmetadata = start_detection;
    }
```

Listening to an event

- Two ways:

```
video.onloadedmetadata = start_detection;

video.addEventListener("loadedmetadata", start_detection);
```

In the first case, be careful not to write

```
video.onloadedmetadata = start_detection();
```

This would execute the function, rather than hooking it to an event!

Detecting video size

- □ What’s de difference between .width, .clientWidth and .videoWidth? (the same for height)

```
function start_detection(event){
        var video = document.getElementById("myvideo");
        console.log("size: ", video.width, video.height);
        console.log("client size: ", video.clientWidth, video.clientHeight);
        console.log("video size: ", video.videoWidth, video.videoHeight);
}
```

Output:

```
size:  0 0
client size:  640 480
video size:  640 480
```

Sizes

- width is the size specified in the HTML tag, if any. It can be modified to resize the video. If not specified, it is 0
- videoWidth is the original video width (read-only)
- clientWidth is the width at which the video is currently shown in the web page (read-only)

Sizes

```
function start_detection(event){
        var video = document.getElementById("myvideo");
        video.width = 300;
        console.log("size: ", video.width, video.height);
        console.log("client size: ", video.clientWidth, video.clientHeight);
        console.log("video size: ", video.videoWidth, video.videoHeight);
}
```

Output:

```
size:  300 0
client size:  300 225
video size:  640 480
```

Video to canvas

- JSARToolKit works on canvas, thus we must copy the video frames in the canvas

```
function start_detection(event){
       var video = document.getElementById("myvideo");
       var canvas = document.getElementById("mycanvas");
       var ctx = canvas.getContext("2d");
       canvas.width = video.clientWidth;
       canvas.height = video.clientHeight;
       ctx.drawImage(video,0,0, canvas.width, canvas.height);
```

- Assign the canvas the same size of displayed video
- Copy the image **and rescale it**
- Check what happens with just

```
ctx.drawImage(video,0,0);
```

Canvas update

- The canvas must be updated at each frame
- Try the `timeUpdate` video event

```
function start_detection(event){
        var video = document.getElementById("myvideo");
        var canvas = document.getElementById("mycanvas");
        var ctx = canvas.getContext("2d");
        canvas.width = video.clientWidth;
        canvas.height = video.clientHeight;

        video.ontimeupdate = function(){
            ctx.drawImage(video,0,0,canvas.width,canvas.height);
        }
}
```

Canvas update

- `ontimeupdate`: poor performance
- Alternative solution

```
setInterval(function(){
      ctx.drawImage(video,0,0,canvas.width, canvas.height);
}, 40);
```

- First parameter: function to be called every n milliseconds

```
setInterval(function(){
      ctx.drawImage(video,0,0, canvas.width, canvas.height);
}, 40);
```

- Second parameters: the time interval (why 40 ms?)

```
setInterval(function(){
      ctx.drawImage(video,0,0, canvas.width, canvas.height);
}, 40);
```

Hide the video

- At this point we have two copies of the video on screen. We can hide the original video and show only the canvas

```
function start_detection(event){
    var video = document.getElementById("myvideo");
    var canvas = document.getElementById("mycanvas");
    var ctx = canvas.getContext("2d");
    canvas.width = video.clientWidth;
    canvas.height = video.clientHeight;
    video.style.display = “none”;

    setInterval(function(){
        ctx.drawImage(video,0,0,canvas.width, canvas.height);
    }, 40);
}
```

Setting up JSARToolKit

- JSARToolKit uses raster objects to read data from canvas

```
function start_detection(event){
      (…)
      // setup JSARToolKit
      var raster = new NyARRgbRaster_Canvas2D(canvas);
```

Setting up JSARToolKit

- Params are used to properly initialize the camera model with the correct aspect ratio

```
function start_detection(event){
      (…)
      // setup JSARToolKit
      var raster = new NyARRgbRaster_Canvas2D(canvas);
      var param = new FLARParam(canvas.width,canvas.height);
```

Setting up JSARToolKit

- The detector is the JSARToolKit core. It detects multiple markers simultaneously

```
function start_detection(event){
      (…)
      // setup JSARToolKit
      var raster = new NyARRgbRaster_Canvas2D(canvas);
      var param = new FLARParam(canvas.width,canvas.height);
      var detector = new FLARMultiIdMarkerDetector(param,76);
```

- 76 is the real size of the markers to be detected, in millimeters

This real-world size is important for detection. Can you figure out why?

Setting up JSARToolKit

- Tells the detector to continuously track markers across multiple frames

```
function start_detection(event){
        (…)
        // setup JSARToolKit
        var raster = new NyARRgbRaster_Canvas2D(canvas);
        var param = new FLARParam(canvas.width,canvas.height);
        var detector = new FLARMultiIdMarkerDetector(param,76);
        detector.setContinueMode(true);
```

Processing each frame

- □ Now process each frame. We set canvas.changed = true to tell JSARToolKit that the canvas has changed and must be processed

```
setInterval(function(){
       ctx.drawImage(video,0,0,canvas.width, canvas.height);
       canvas.changed = true;
}, 40);
```

Processing each frame

- □ Run the detector. The second parameter (128) is a **threshold** (more about this later)
- □ Returns the number of detected markers

```
setInterval(function(){
       ctx.drawImage(video,0,0,canvas.width, canvas.height);
       canvas.changed = true;
       var markerCount = detector.detectMarkerLite(raster,128);
}, 40);
```

Processing each frame

- □ If at least a maker has been detected, get the ***transformation matrix*** of the first one

```
setInterval(function(){
        ctx.drawImage(video,0,0,canvas.width, canvas.height);
        canvas.changed = true;
        var markerCount = detector.detectMarkerLite(raster,128);
        if(markerCount > 0){
                var tmat = new NyARTransMatResult();
                detector.getTransformMatrix(0, tmat);
                console.log(tmat);
        }
}, 40);
```

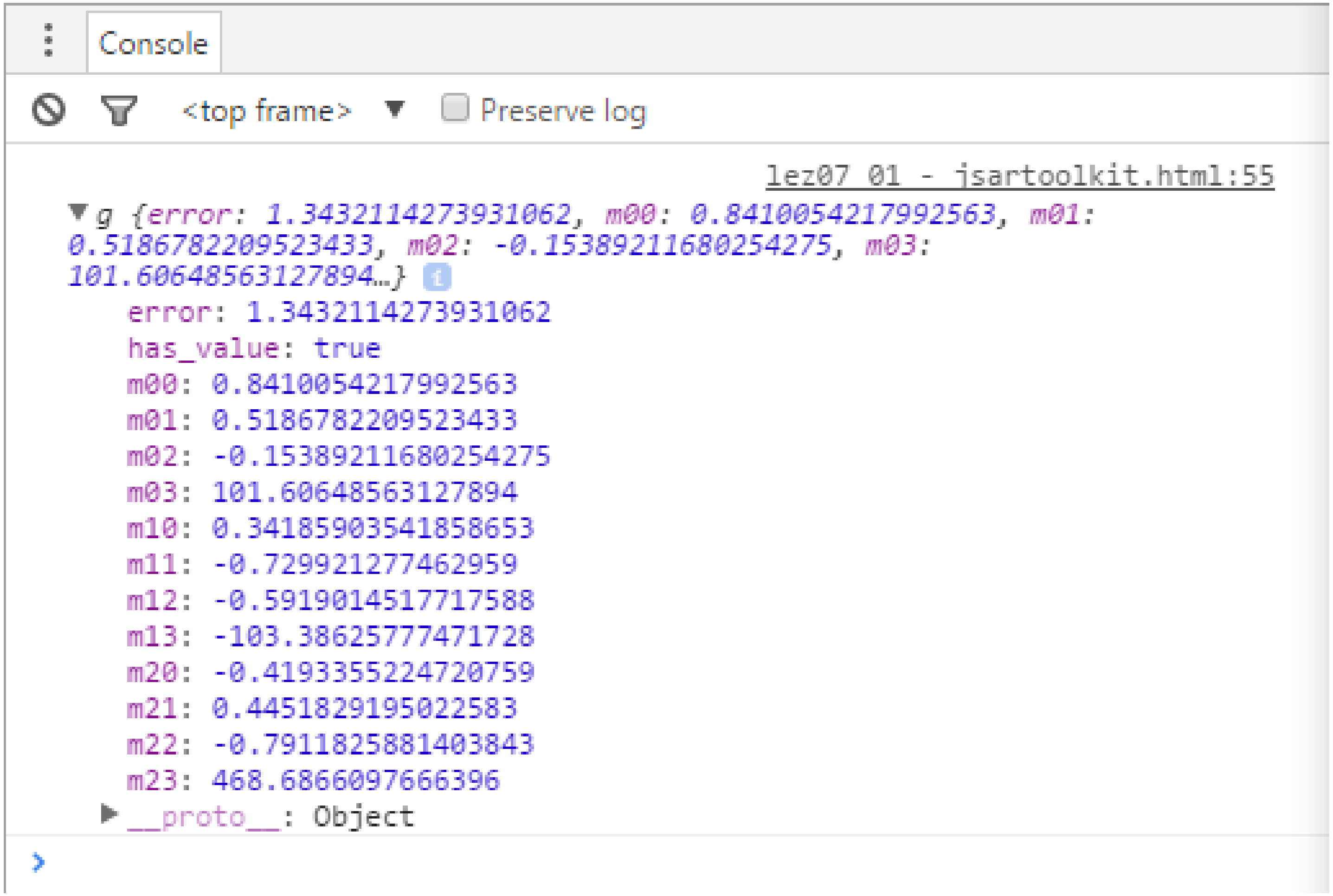
Final start_detection() code

```
function start_detection(event){
    // canvas setup
    var video = document.getElementById("myvideo");
    var canvas = document.getElementById("mycanvas");
    var ctx = canvas.getContext("2d");
    canvas.width = video.clientWidth;
    canvas.height = video.clientHeight;
    video.style.display = "none";

    // JSARToolKit setup
    var raster = new NyARRgbRaster_Canvas2D(canvas);
    var param = new FLARParam(canvas.width, canvas.height);
    var detector = new FLARMultiIdMarkerDetector(param, 76);
    detector.setContinueMode(true);

    // frame loop
    setInterval(function(){
        ctx.drawImage(video,0,0,canvas.width,canvas.height);
        canvas.changed = true;
        var markerCount = detector.detectMarkerLite(raster,128);
        if(markerCount > 0){
            var tmat = new NyARTransMatResult();
            detector.getTransformMatrix(0, tmat);
            console.log(tmat);
        }
    }, 40);
}
```

Results



```
                                              lez07_01 - jsartoolkit.html:55
▼g {error: 1.3432114273931062, m00: 0.8410054217992563, m01:
0.5186782209523433, m02: -0.15389211680254275, m03:
101.60648563127894…}
    error: 1.3432114273931062
    has_value: true
    m00: 0.8410054217992563
    m01: 0.5186782209523433
    m02: -0.15389211680254275
    m03: 101.60648563127894
    m10: 0.34185903541858653
    m11: -0.729921277462959
    m12: -0.5919014517717588
    m13: -103.38625777471728
    m20: -0.4193355224720759
    m21: 0.4451829195022583
    m22: -0.7911825881403843
    m23: 468.6866097666396
  ▶__proto__: Object
>
```

Understanding the results

- The transformation matrix represents the marker 3D position respect to the camera
- We will use this data to align a 3D model with the marker
- More info on transformation matrices in the next lessons

Extra: debug canvas

- JSARToolKit allows the visualization of a debug canvas where you can see the internal processing
- Add a canvas with **`id=“debugCanvas”`** in your HTML code (must have this name!)
- Add this javascript code after JSARToolKit initialization

```
// JSARToolKit debugging
var dcanvas = document.getElementById("debugCanvas");
dcanvas.width = canvas.width;
dcanvas.height = canvas.height;
DEBUG = true;
```

Extra: adjusting the threshold

- Threshold values are in the range 0 – 255
- The final result is heavily influenced by the choice of threshold
- Homework: add a slider to your HTML page to change the threshold dynamically
  - Hint: learn to use dat.GUI:
  - `https://github.com/dataarts/dat.gui`

Extra: extracting marker IDs

- Each marker has its own ID encoded in the marker shape (numbers from 0 to 99)
- Extracting the marker ID could be useful (e.g. in a multi-marker system)
- Check the code:

`http://www.html5rocks.com/en/tutorials/webgl/jsartoolkit_webrtc/`