



Lectures

Sérgio M. Rebelo

srebelo@dei.uc.pt



Jessica Parente

anatr@dei.uc.pt



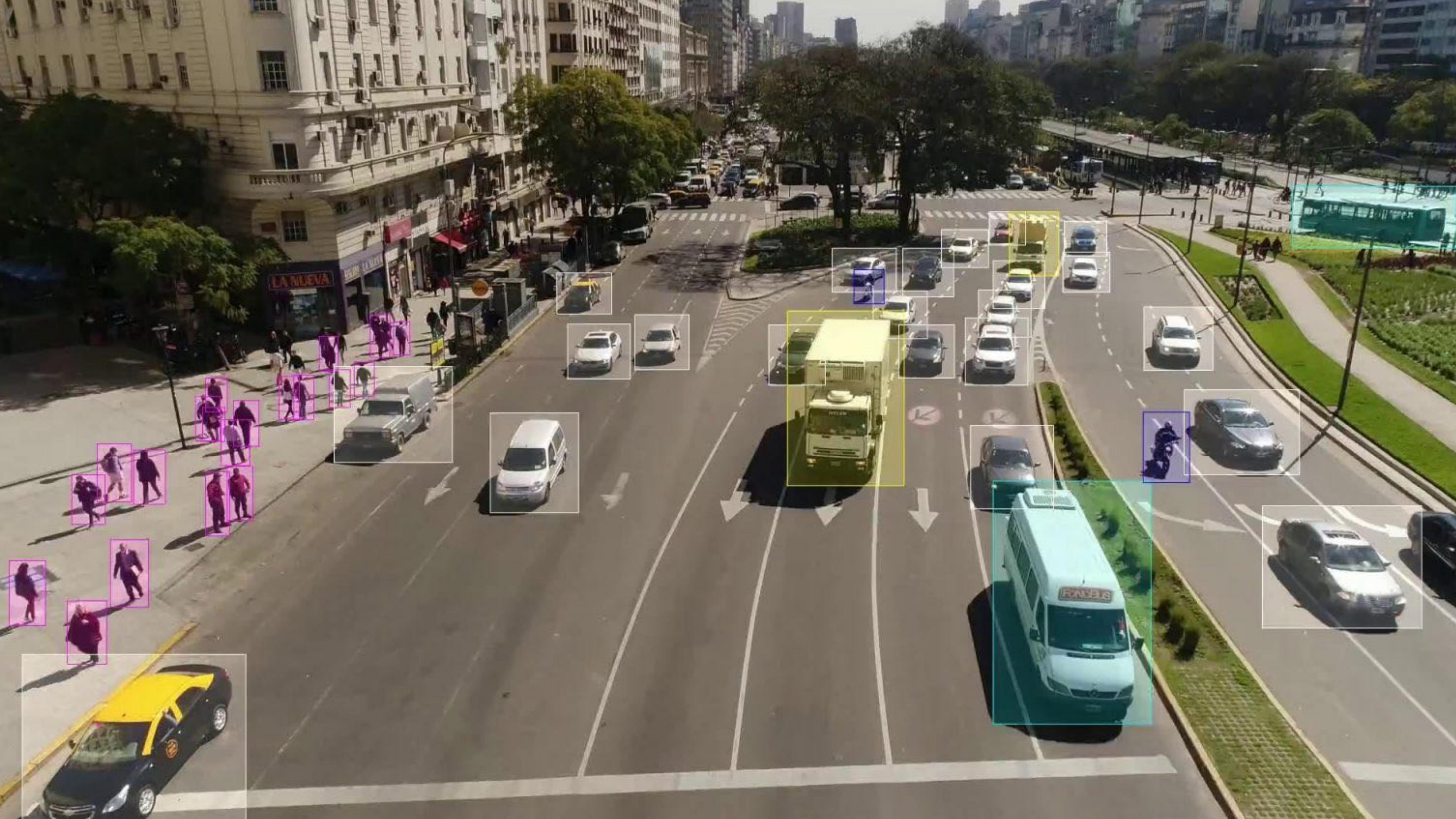
Introduction to Computer Vision

Introduction to Computer Vision for Creative Coding











<https://processing.org/download>

The screenshot shows a web browser window displaying the Processing.org website. The page is titled "Video" and contains information about the Video library, its source code on GitHub, and a table of methods for the Capture datatype. The "Capture" section is highlighted with a light gray background.

Video

The Video library plays movie files and captures video data from a camera. Video can be captured from USB Cameras, IEEE 1394 (Firewire) Cameras, or Video Cards with composite or S-video input devices connected to the computer. Movies can be loaded from files located on your computer or anywhere on the Internet. It is based on the GStreamer multimedia framework, and uses the gstreamer-java bindings to interface GStreamer from Java to support a wide range of media formats. We recommend using H.264, but many other formats will work as well.

The source code is available on the [processing-video GitHub repository](#). Please report bugs here.

Capture

<code>Capture</code>	Datatype for storing and manipulating video frames from an attached capture device such as a camera.
<code>available()</code>	Returns "true" when a new frame from the device is available to read.
<code>frameRate()</code>	Sets how often frames are read from the capture device.
<code>read()</code>	Reads the current frame of the device.
<code>start()</code>	Starts capturing frames from an attached device.

A screenshot of a web browser displaying the p5.js.org website. The page title is "Video Capture". On the left, there is a vertical navigation menu with links: Home (highlighted), Editor, Download, Donate, Get Started, Reference, Libraries, Learn, Teach, Examples (highlighted), and Contribute. A large red graphic element is positioned next to the navigation menu. The main content area features a large black video feed placeholder. Below the video feed are three buttons: "run", "reset", and "copy". At the bottom of the code editor, the line of code "let capture;" is visible. The browser's address bar shows "p5js.org". The top right corner of the browser window displays the "Processing Foundation" logo.

Processing p5.js Processing.py Processing for Android Processing for Pi

Processing Foundation

English Español 简体中文 한국어 हिन्दी

p5.js

Video Capture

< Back to Examples

Capture video from the webcam and display on the canvas as well with invert filter. Note that by default the capture feed shows up, too. You can hide the feed by uncommenting the `capture.hide()` line.

run reset copy

```
let capture;
```

<https://p5js.org/examples/dom-video-capture.html>

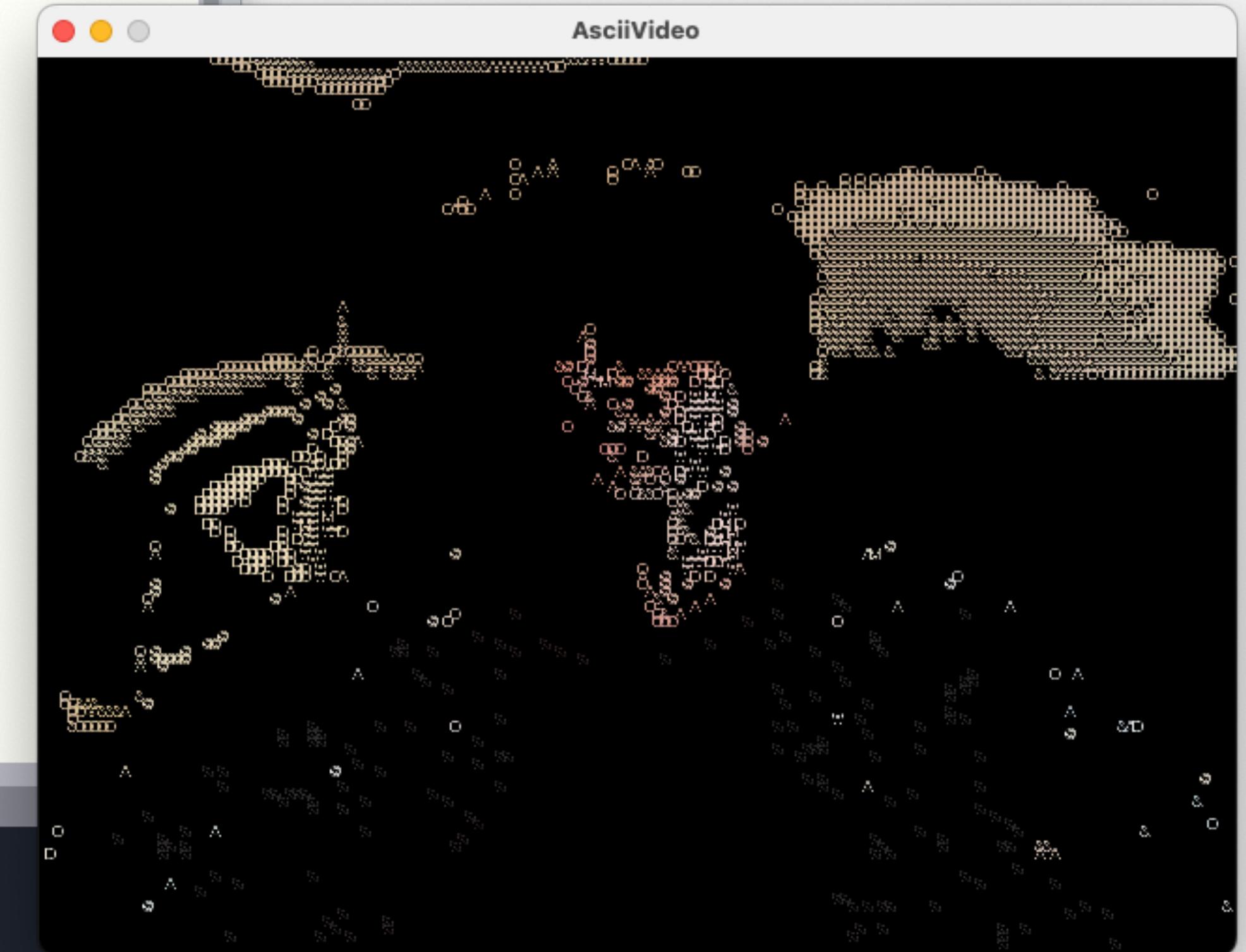
AsciiVideo | Processing 4.2

AsciiVideo

```
9  */
10
11 import processing.video.*;
12
13 Capture video;
14 boolean cheatScreen;
15
16 // Characters sorted according to their visual density
17 String letterOrder =
18 " .-_':;^=+/\\"|)\{\>)iv\xclrs{*\}!?\][ltaeo/zjLU" +
19 "nT#JCwfy325Fp6mqSghVd4EgXPGZbYkOA&8U$@KHDBWNMR0Q";
20 char[] letters;
21
22 float[] bright;
23
24 PFont font;
25 float fontsize = 1.5;
26
27
28 void setup() {
29   size(640, 480);
30
31   // This the default video input, see the GettingStartedCapture
32   // example if it creates an error
33   video = new Capture(this, 160, 120);
34
35   // Start capturing the images from the camera
36   video.start();
37
38   int count = video.width * video.height;
```

Java ▾

Console Errors



Live Video

The screenshot shows the Processing 4.2 IDE interface. The title bar reads "VideoCapture | Processing 4.2". The main window displays the following Java code:

```
VideoCapture  
Capture cam;  
void setup() {  
    size(1280, 720);  
    String[] cameras = Capture.list();  
    if (cameras.length == 0) {  
        println("There are no cameras available for capture.");  
        exit();  
    } else {  
        println("Available cameras:");  
        printArray(cameras);  
    }  
    cam = new Capture(this, 1280, 720, cameras[0], 30);  
    cam.start();  
}  
frameRate(30);  
void draw() {  
    if (cam.available() == true) {  
        cam.read();  
        image(cam, 0, 0);  
    }  
}
```

The code uses the `Capture` class from the `Video library`. The `list()` method returns an array of strings representing available cameras. The `available()` method checks if a camera is ready for use, and `read()` captures a frame.

In the bottom right corner of the code editor, there is a small circular icon with a blue "86" and a "Java" dropdown menu.

The console window at the bottom shows the output of the code execution:

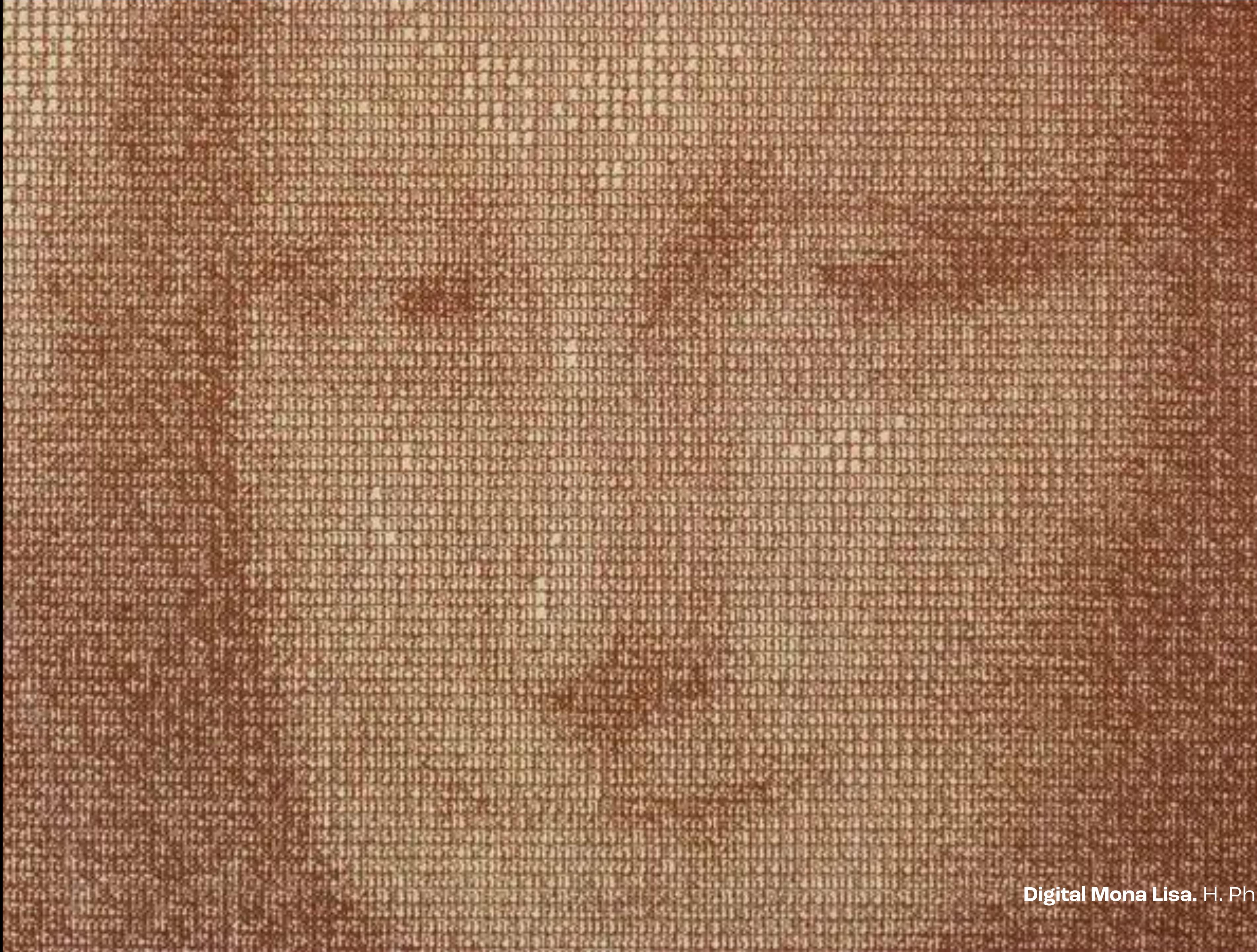
```
Auto Format finished.  
Scanning GStreamer plugins... Done.  
Available cameras:  
[0] "FaceTime HD Camera"  
[1] "Sérgio Rebelo Camera"
```

At the bottom left, there are tabs for "Console" and "Errors".

Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Boilerplates/VideoCaptureBoilerplate>

or Video library examples → Capture → GettingStartedCapture.

Software Mirrors

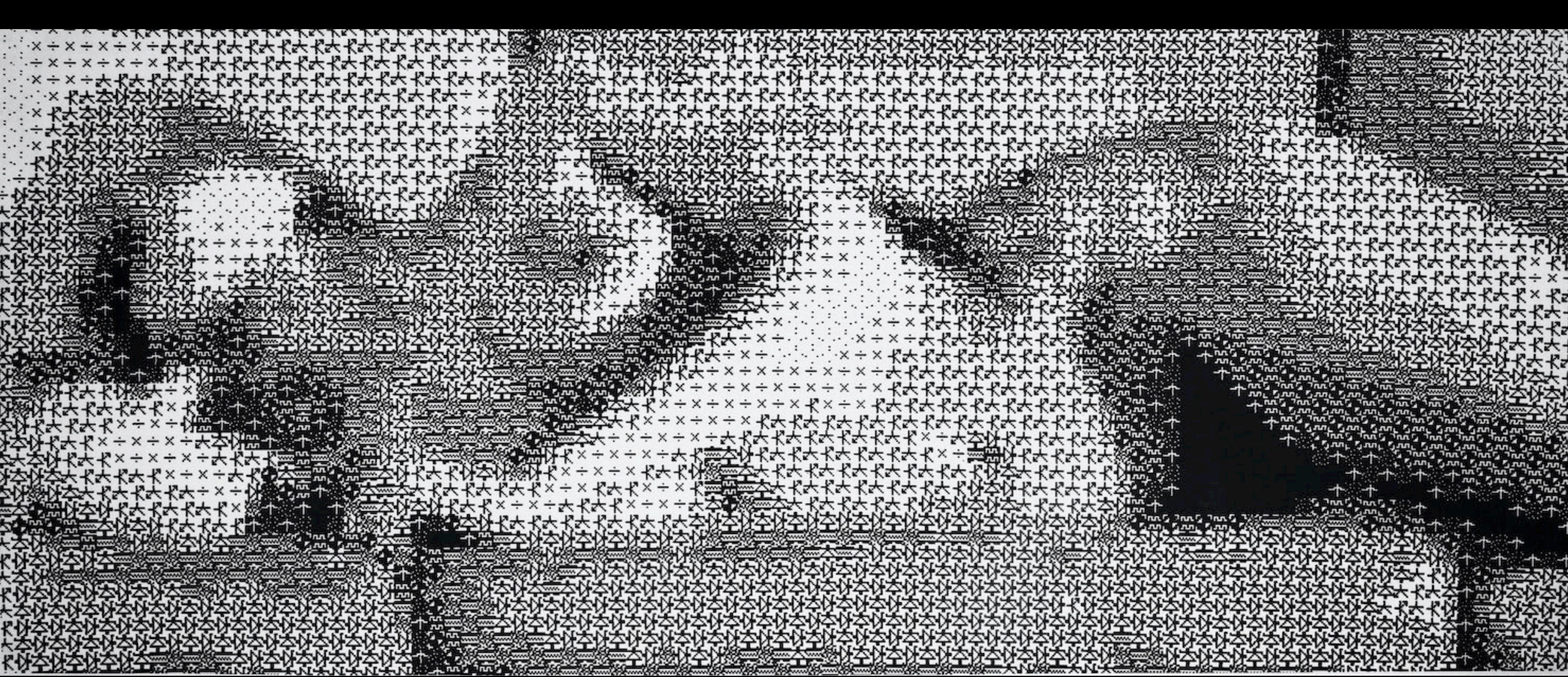


Digital Mona Lisa. H. Philip Peterson (1965)



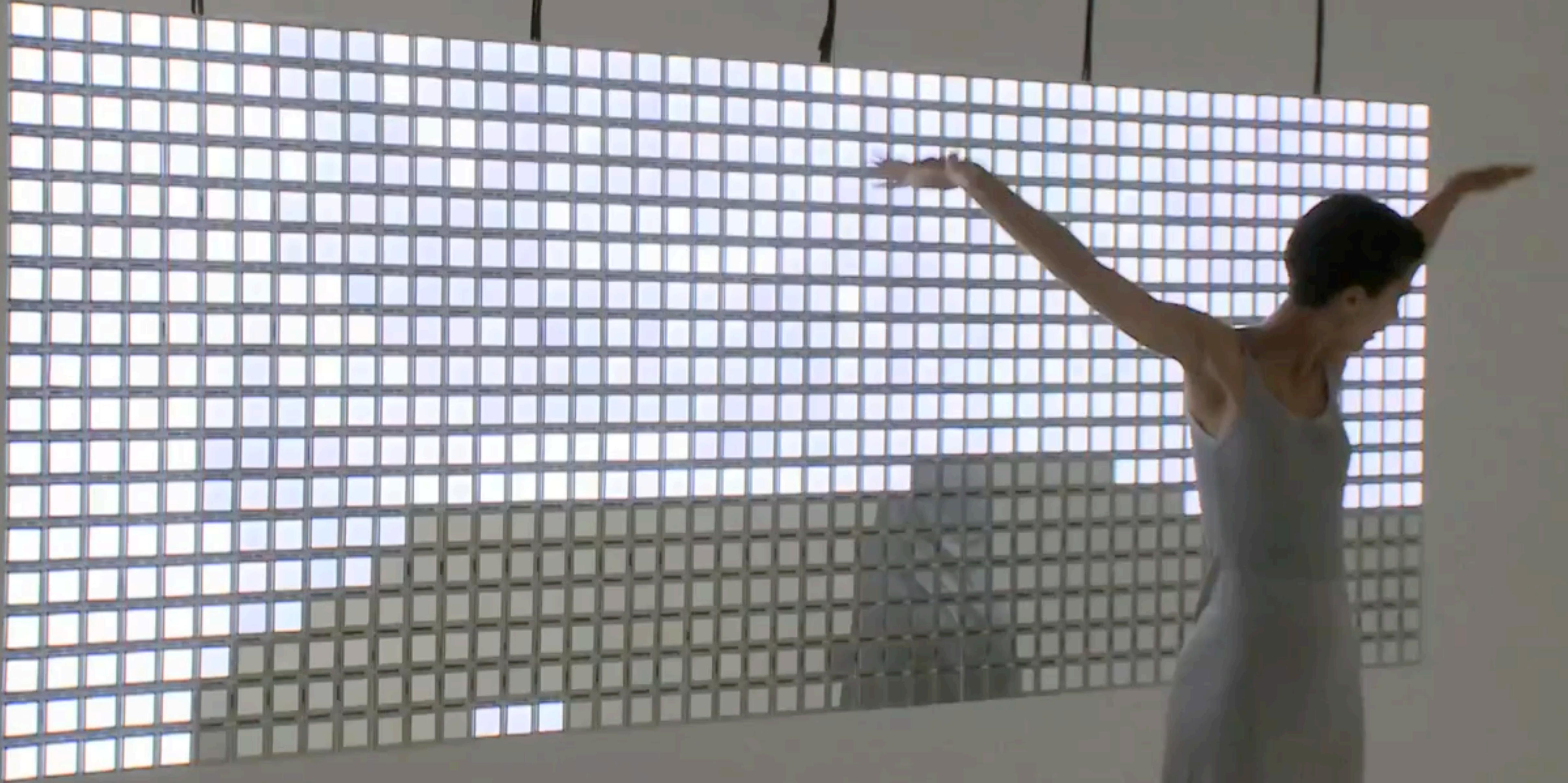
Studies in Perception.

Kenneth C. Knowlton and Leon D. Harmon,
BELFIX. (1968)



Studies in Perception.

Kenneth C. Knowlton and Leon D. Harmon,
BELFIX. (1968)





Peg Mirror.
Daniel Rosin (2007)

Photomaton.

Sérgio M. Rebelo, Tiago Martins, et al. (2020)



Magro, de olhos azuis, carão moreno, / Bem servido de pés, meão na altura, / Triste de facha, o mesmo de figura, / Nariz alto no meio, e não pequeno; // Incapaz de assistir num só terreno / Mais propenso ao furor do que à ternura, / Bebendo em níveis mãos por taça escura / De zelos infernais letal veneno;



Fundação
ERNESTO DE
SOUZA BRAGA

Português - 2015/01/15 - 1000

www.digitais.ufpb.br/erste



José maria nicolau fugiu. Quem o apanha? / Nunca ele pedalou tanto como agora / Decerto vai chegar antes da hora / A etapa era decisiva e está ganha // Ele que várias vezes deu a volta a portugal / deu desta vez a volta a quê? Talvez à vida



Fundação
ERNESTO DE
SOUZA BRAGA

Português - 2015/01/15 - 1604

www.digitais.ufpb.br/erste



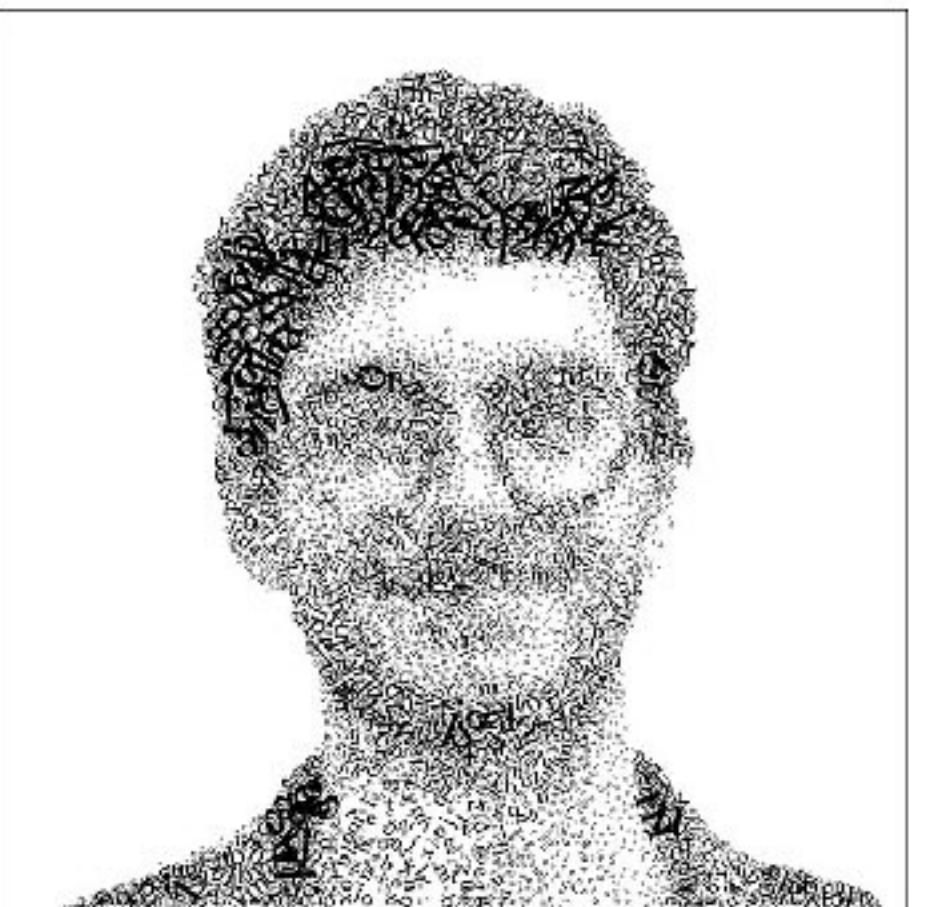
Só! / Ai do Lusiada, coitado, / Que vem de tão longe, coberto de pó. / Que não ama, nem é amado, / Lúgubre Outono, no mês de Abril! / Que triste foi o seu fado! / Antes fosse pra soldado, / Antes fosse pró Brasil...



Fundação
ERNESTO DE
SOUZA BRAGA

Português - 2015/01/15 - 1000

www.digitais.ufpb.br/erste



Não sei porquê, acordei com este poema na cabeça // Olha, Daisy: quando eu morrer tu hás de / dizer aos meus amigos af de Londres, / embora não o sintas, que tu escondes / a grande dor da minha morte.

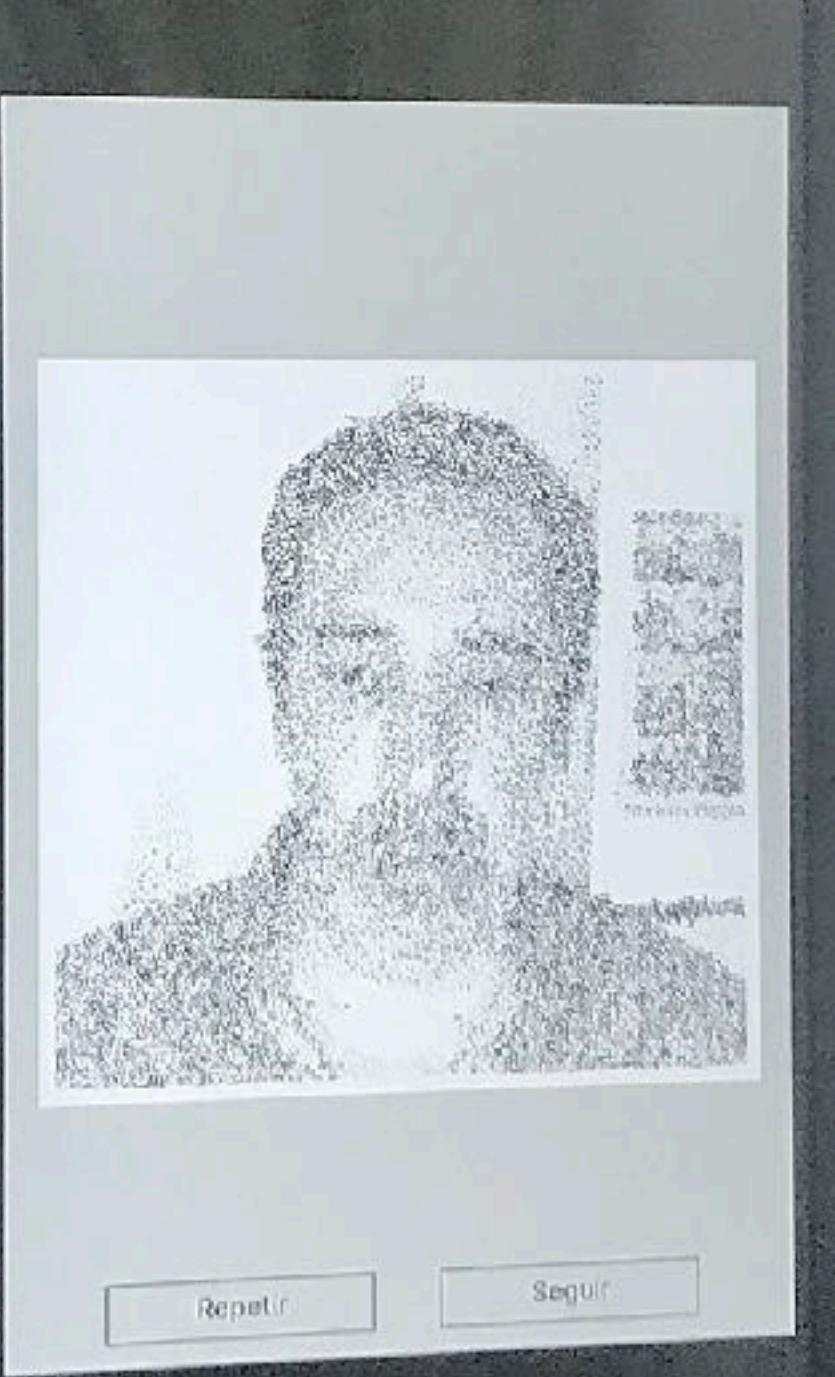


Fundação
ERNESTO DE
SOUZA BRAGA

Português - 2015/01/15 - 1448

www.digitais.ufpb.br/erste





The image shows the Processing 4.2 IDE interface. The top window is titled "black_and_white | Processing 4.2". The code editor contains the following Java code:

```
54
55     for ( int i = 0; i < cols; i++) {
56         //begin loop for rows
57         for ( int j = 0; j < rows; j++) {
58             int x = i * cellsize + cellsize/2;
59             int y = j * cellsize + cellsize/2;
60             int loc = int(x + y * frame.width);
61
62             loc = constrain(loc, 0, frame.pixels.length-1);
63
64             noStroke();
65             //make a new color with alpha value
66             color c = color(red(frame.pixels[loc]), green(frame.pi
67
68             //our drawing code, we are using translate
69             pushMatrix();
70             translate(x, y);
71
72             //each rectangle colored white with a size determined
73             fill(brightness(c));
74             float side = ((brightness(c)/255.0f)*cellsize);
75
76             rect(0, 0, side, side);
77             nonMatrix();
```

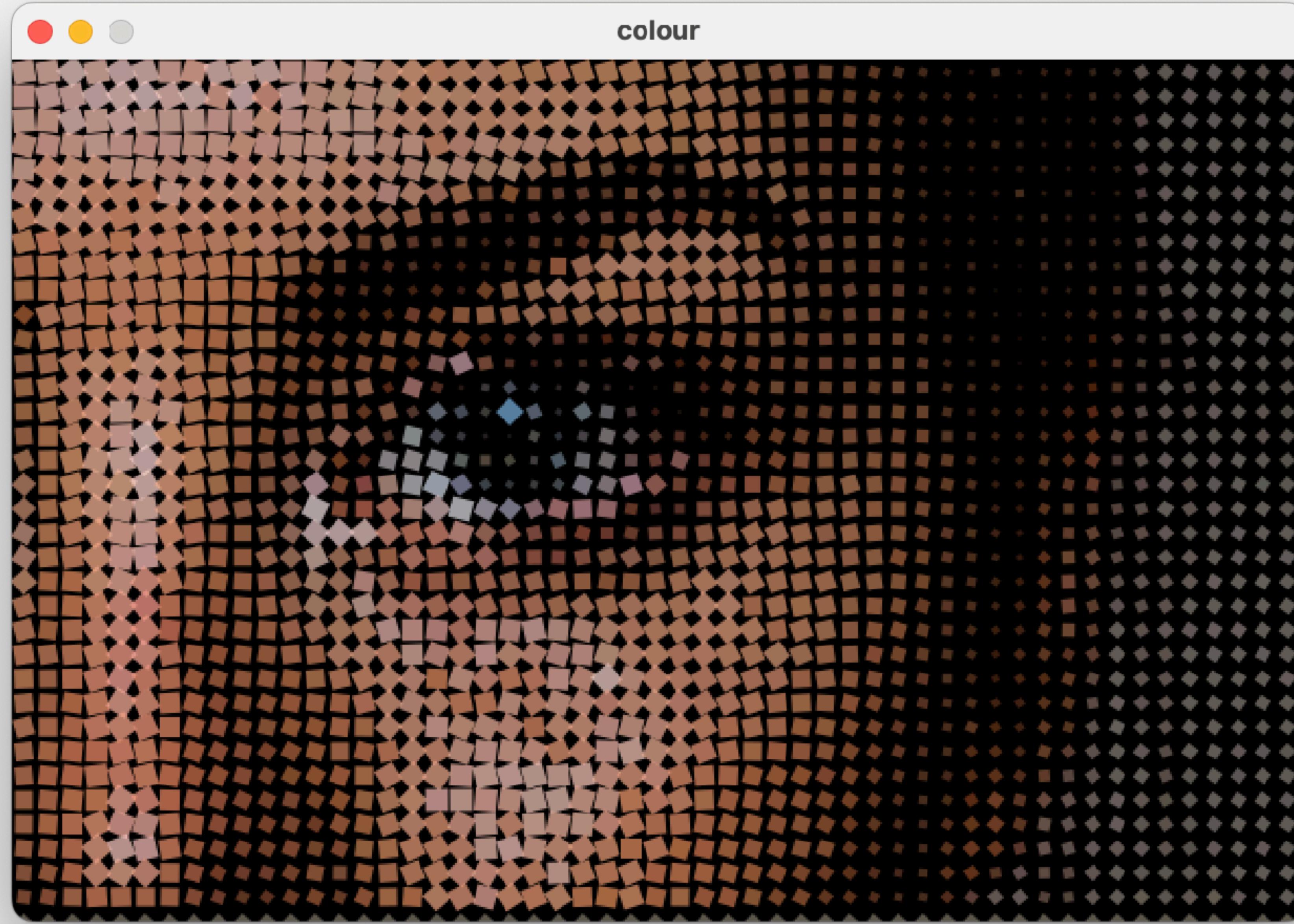
The bottom window shows the output of the sketch, titled "black_and_white". It displays a black and white halftone pattern of a robot's head and shoulders, rendered in a grid of pixels.

Processing video library using bundled GStreamer 1.20.3
Scanning GStreamer plugins... Done.

Console Errors

Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/1-Mirrors/BlackAndWhite>





Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/1-Mirrors/Colour>

Video Tracking

Brightness

The screenshot shows the Processing 4.2 IDE interface. The title bar reads "BrightnessTracking | Processing 4.2". The main window displays the code for the "BrightnessTracking" sketch. The code uses the GStreamer library to capture video from a camera and create a brightness map. It iterates through each pixel of the frame, determines its brightness, and updates a PImage object "m" with the color of the brightest pixel found. A preview window on the right shows a grayscale close-up of a person's eye, which is the subject of the brightness tracking. The status bar at the bottom indicates "Processing video library using bundled GStreamer 1.20.3" and "Scanning GStreamer plugins... Done.". The bottom navigation bar includes "Console" and "Errors" tabs.

```
// frame = cam.copy();

// brightness map
m = new PImage(frame.width, frame.height, ARGB)

PVector brightest = new PVector (-1, -1);
float max = -1; // Brightness of the brightest

m.loadPixels();
for (int y = 0; y < frame.height; y++) {
    for (int x = 0; x < frame.width; x++) {
        int loc = x + y * frame.width;
        // Determine the brightness of the pixel
        float current = brightness(frame.pixels[loc]);

        m.pixels[loc] = color(current);

        if (current > max) {
            max = current;
            brightest.y = y;
            brightest.x = x;
        }
    }
}
```

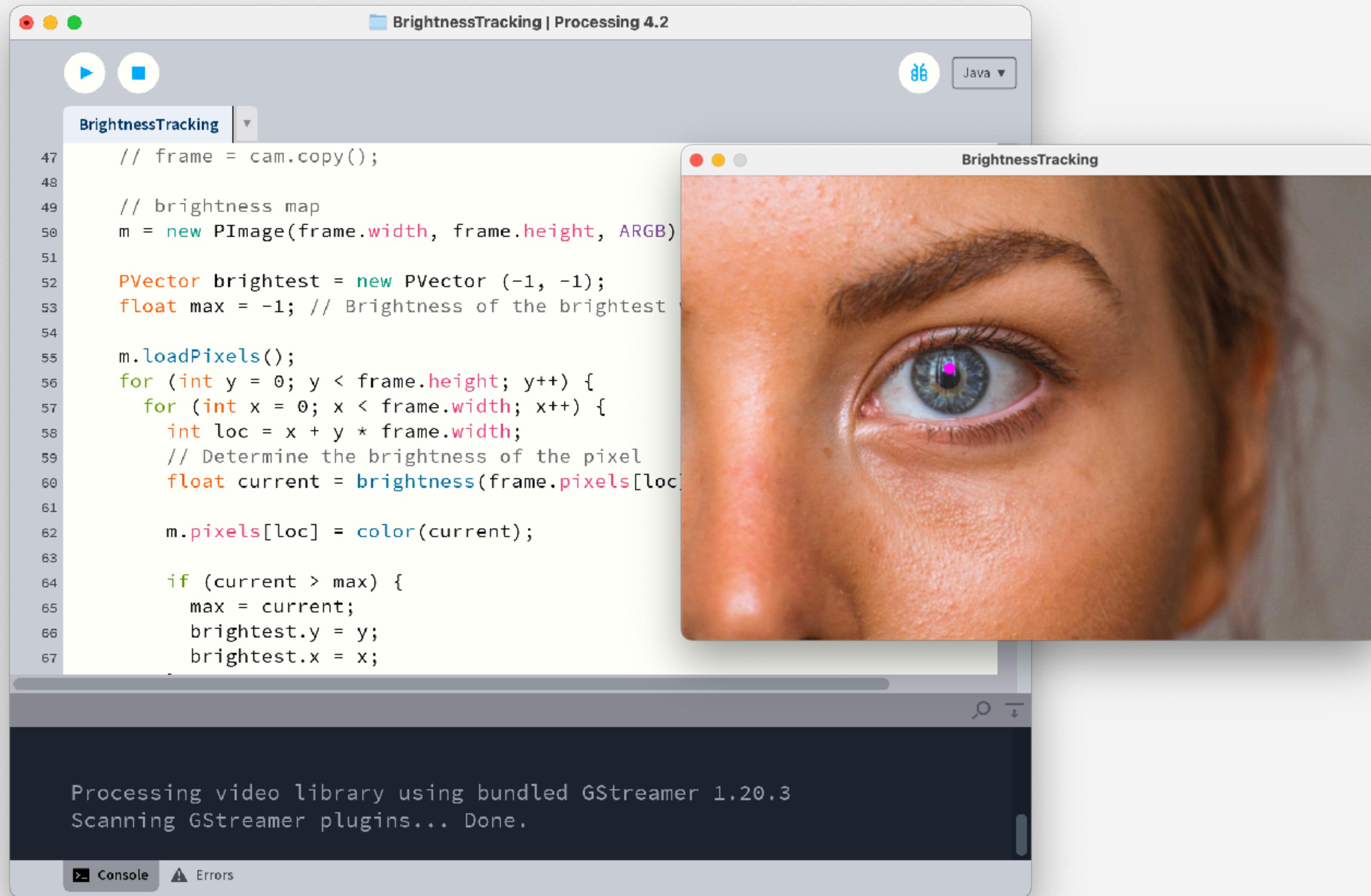
BrightnessTracking

47 // frame = cam.copy();
48
49 // brightness map
50 m = new PImage(frame.width, frame.height, ARGB)
51
52 PVector brightest = new PVector (-1, -1);
53 float max = -1; // Brightness of the brightest
54
55 m.loadPixels();
56 for (int y = 0; y < frame.height; y++) {
57 for (int x = 0; x < frame.width; x++) {
58 int loc = x + y * frame.width;
59 // Determine the brightness of the pixel
60 float current = brightness(frame.pixels[loc]);
61
62 m.pixels[loc] = color(current);
63
64 if (current > max) {
65 max = current;
66 brightest.y = y;
67 brightest.x = x;
68 }
69 }

BrightnessTracking

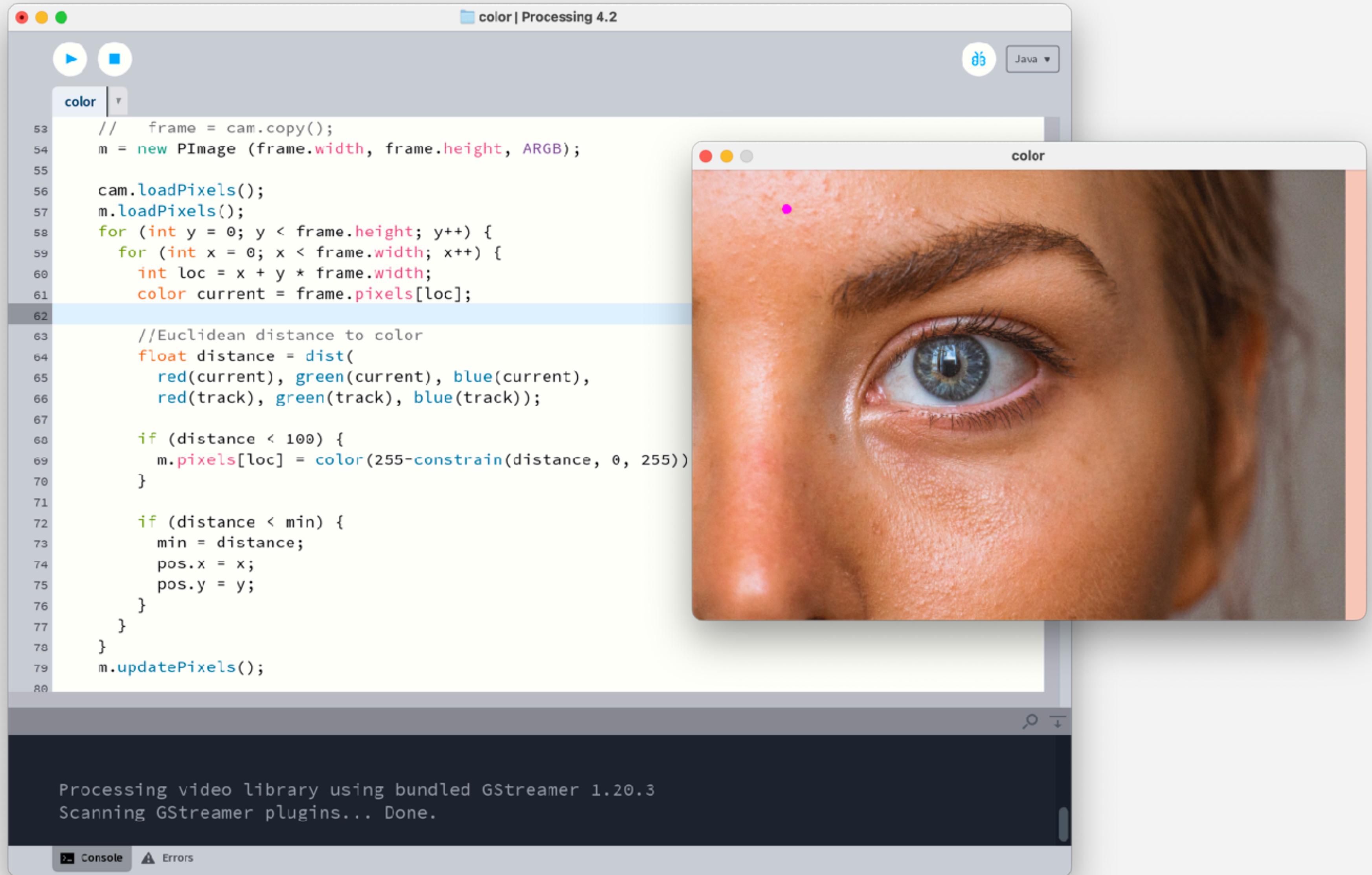
Processing video library using bundled GStreamer 1.20.3
Scanning GStreamer plugins... Done.

Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/2-Tracking/Brightness>



Code: <https://github.com/sengiomrebelo/amazing-robots-cv/tree/main/Demos/2-Tracking/Brightness>

Colour



Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/2-Tracking/Color>

color | Processing 4.2

color

```
53 // frame = cam.copy();
54 m = new PImage (frame.width, frame.height, ARGB);
55
56 cam.loadPixels();
57 m.loadPixels();
58 for (int y = 0; y < frame.height; y++) {
59   for (int x = 0; x < frame.width; x++) {
60     int loc = x + y * frame.width;
61     color current = frame.pixels[loc];
62
63     //Euclidean distance to color
64     float distance = dist(
65       red(current), green(current), blue(current),
66       red(track), green(track), blue(track));
67
68     if (distance < 100) {
69       m.pixels[loc] = color(255-constrain(distance, 0, 255));
70     }
71
72     if (distance < min) {
73       min = distance;
74       pos.x = x;
75       pos.y = y;
76     }
77   }
78 }
79 m.updatePixels();
```

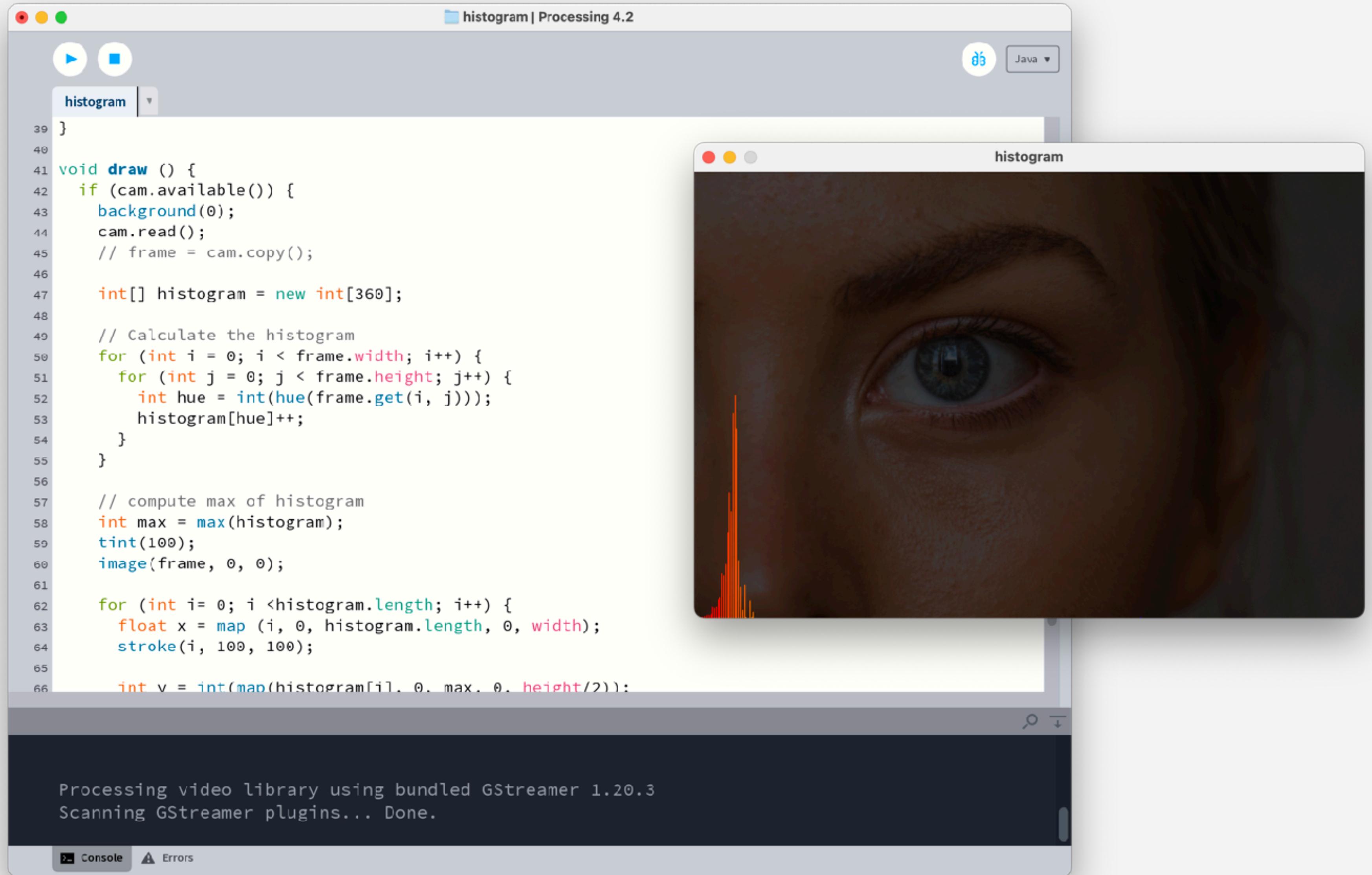
color

Processing video library using bundled GStreamer 1.20.3
Scanning GStreamer plugins... Done.

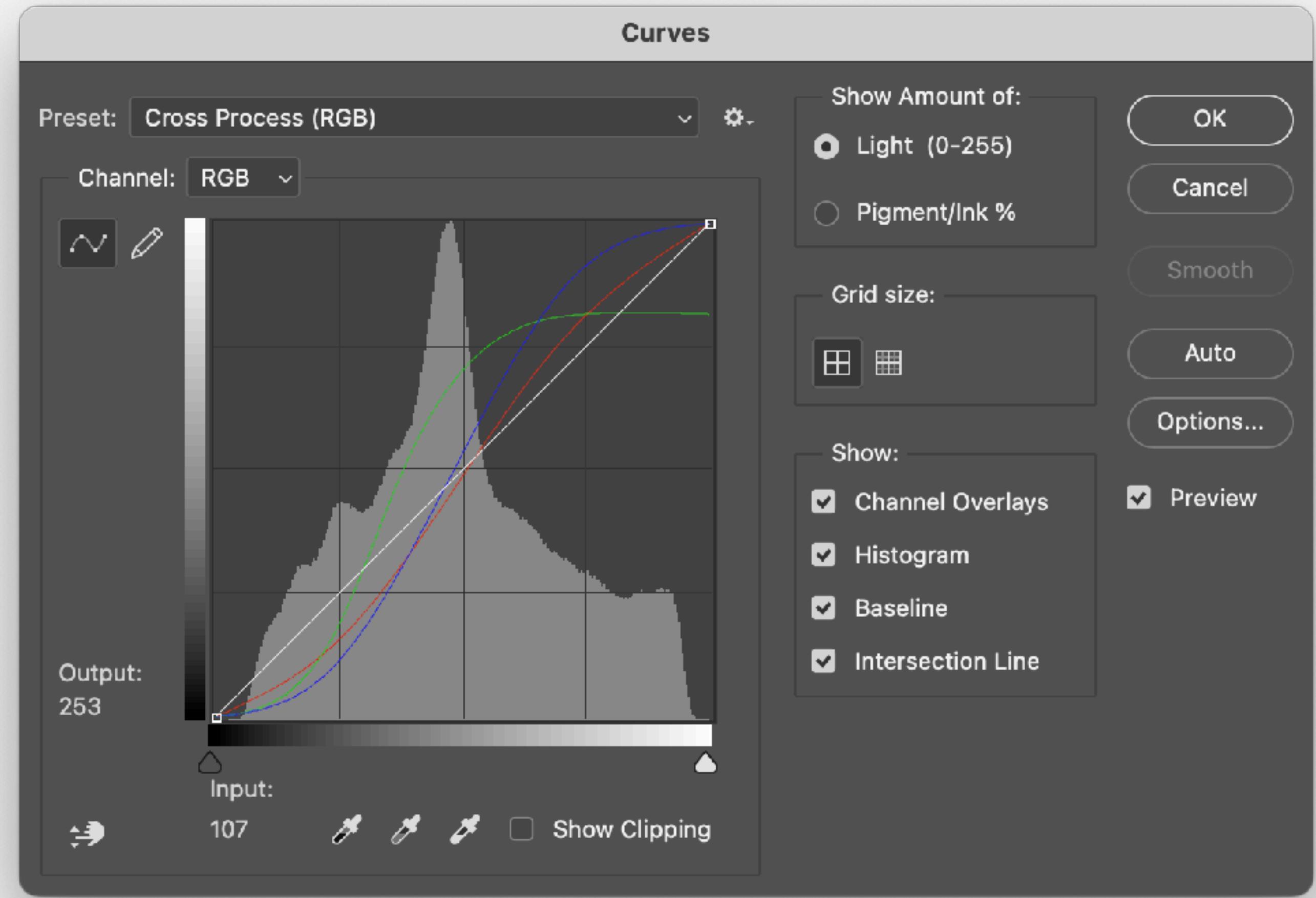
Console Errors

Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/2-Tracking/Color>

Histogram



Code: <https://github.com/sengiomrebelo/amazing-robots-cv/tree/main/Demos/2-Tracking/Histogram>



1526999

Video differencing algorithms

Background Subtraction

backgroundSubtraction | Processing 4.2

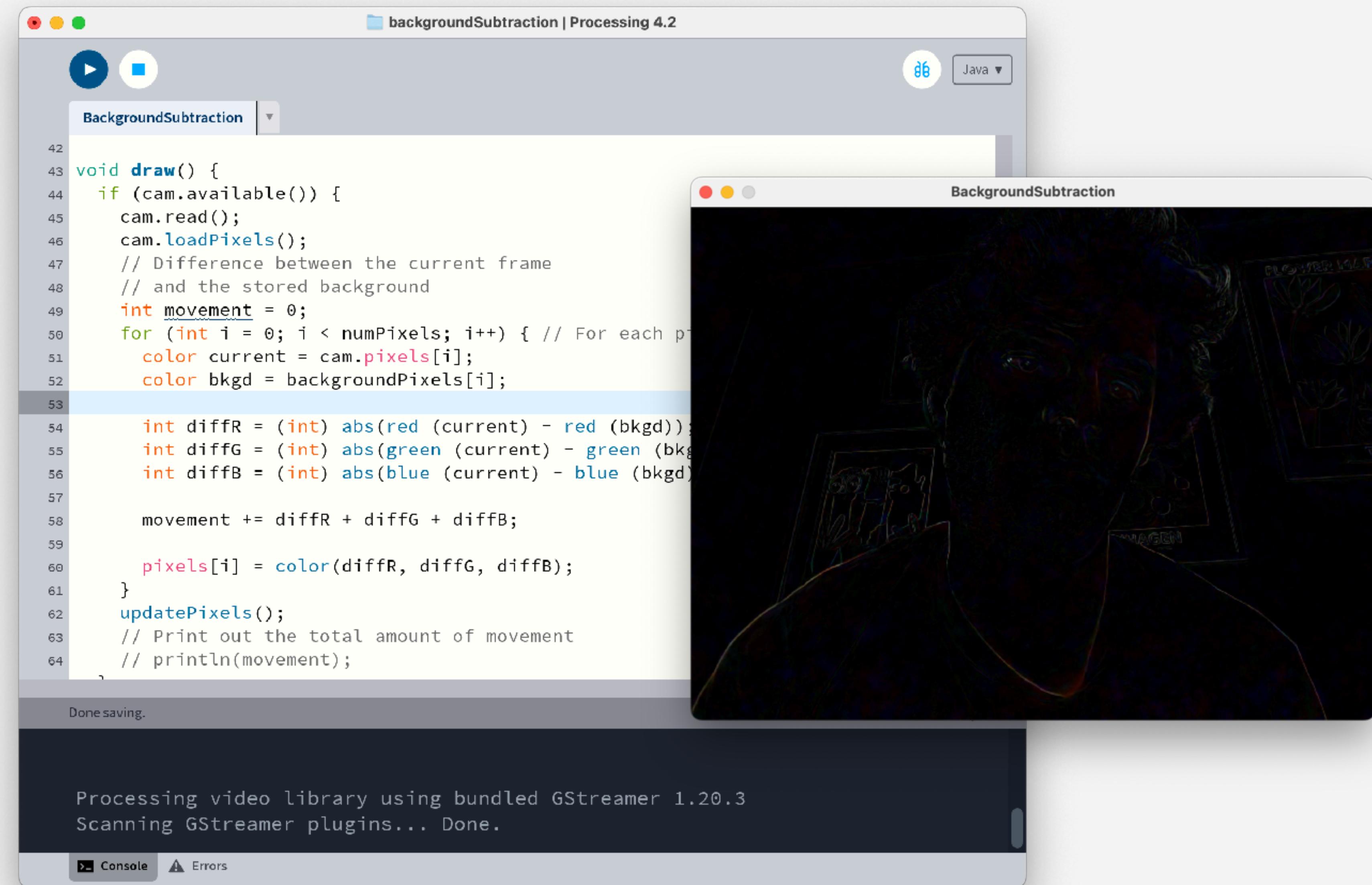
BackgroundSubtraction

```
42
43 void draw() {
44     if (cam.available()) {
45         cam.read();
46         cam.loadPixels();
47         // Difference between the current frame
48         // and the stored background
49         int movement = 0;
50         for (int i = 0; i < numPixels; i++) { // For each p
51             color current = cam.pixels[i];
52             color bkgd = backgroundPixels[i];
53
54             int diffR = (int) abs(red(current) - red(bkgd));
55             int diffG = (int) abs(green(current) - green(bkgd));
56             int diffB = (int) abs(blue(current) - blue(bkgd));
57
58             movement += diffR + diffG + diffB;
59
60             pixels[i] = color(diffR, diffG, diffB);
61         }
62         updatePixels();
63         // Print out the total amount of movement
64         // println(movement);
    }

Done saving.
```

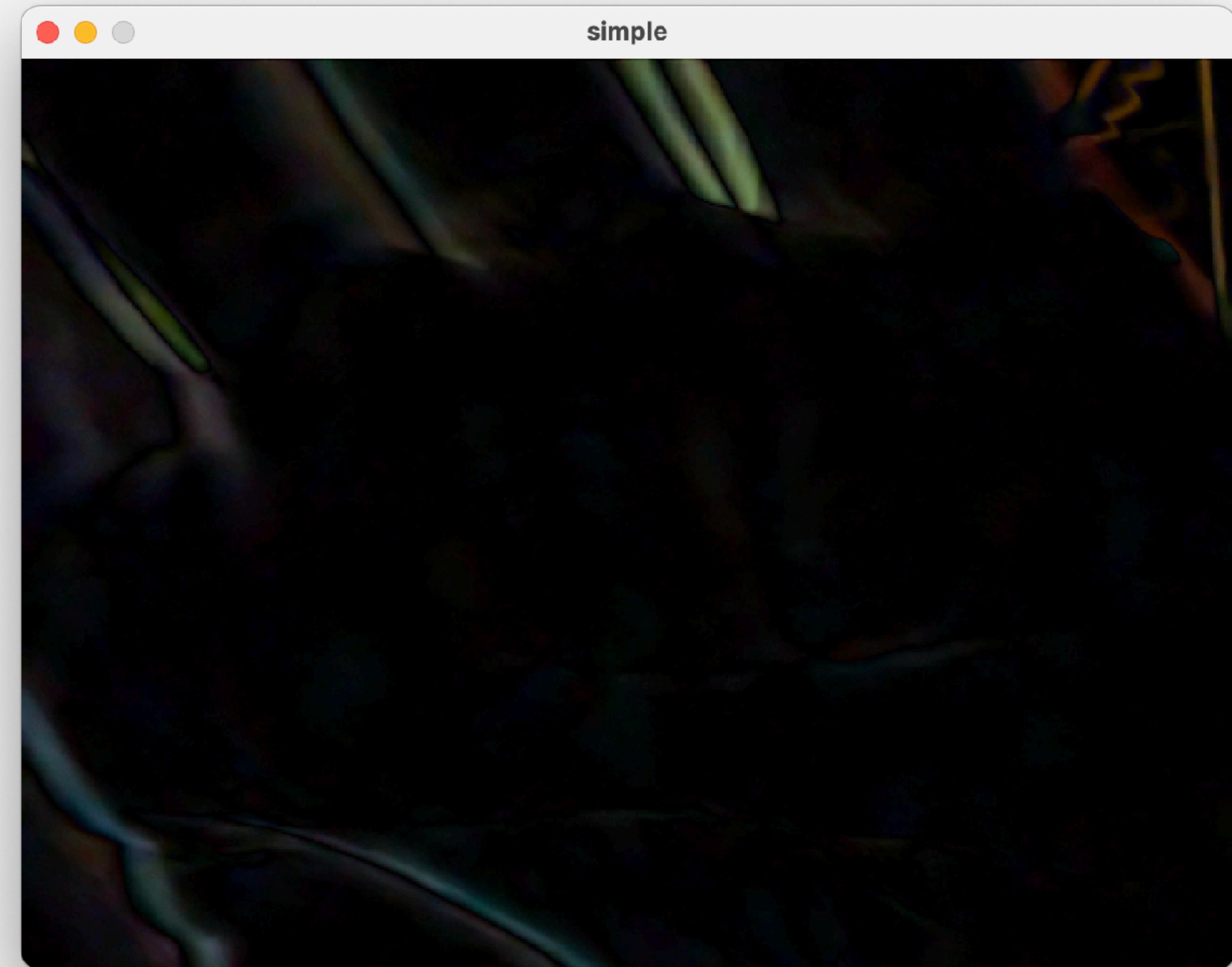
Processing video library using bundled GStreamer 1.20.3
Scanning GStreamer plugins... Done.

Java ▾



Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/3-Differencing/BackgroundSubtraction>

Frame differencing



simple

Frame differencing sketch

The screenshot shows the Processing 4.2 IDE interface. The title bar reads "simple | Processing 4.2". The main area displays the following Java code:

```
33 previous = new PImage (cam.width, cam.height, ARGB);
34 }
35
36 void draw() {
37   if (cam.available()) {
38     background(0);
39     PImage dif = new PImage (cam.width, cam.height, AR
40
41     cam.read();
42     cam.loadPixels();
43     dif.loadPixels();
44
45     for (int i = 0; i < cam.pixels.length; i++) {
46       color current = cam.pixels[i];
47
48       int diffR = (int) abs(red (current) - red (previous));
49       int diffG = (int) abs(green (current) - green (previous));
50       int diffB = (int) abs(blue (current) - blue (previous));
51
52       int movement = diffR + diffG + diffB;
53
54       if (movement > threshold) {
55         dif.pixels[i] = color(diffR, diffG, diffB);
56       }
57     }
58     previous = cam.copy();
59     dif.updatePixels();
60 }
```

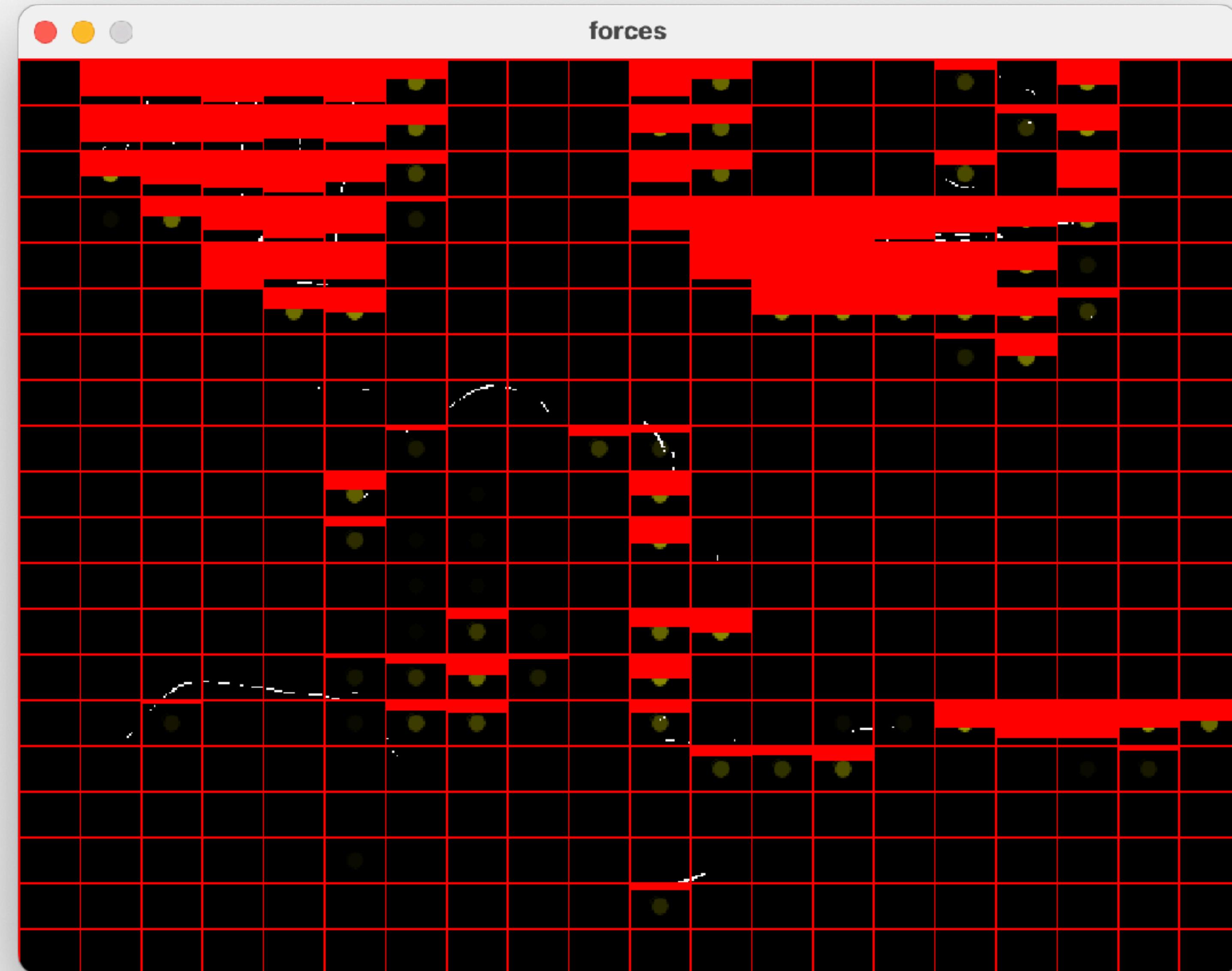
The status bar at the bottom left says "Done saving." and the bottom right says "Processing video library using bundled GStreamer 1.20.3 Scanning GStreamer plugins... Done."

A preview window titled "simple" shows a blurred video feed from a camera, indicating the motion detection effect.

Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/3-Differencing/Frame%20Differencing/Simple>



Code: <https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/3-Differencing/Frame%20Differencing/Forces>



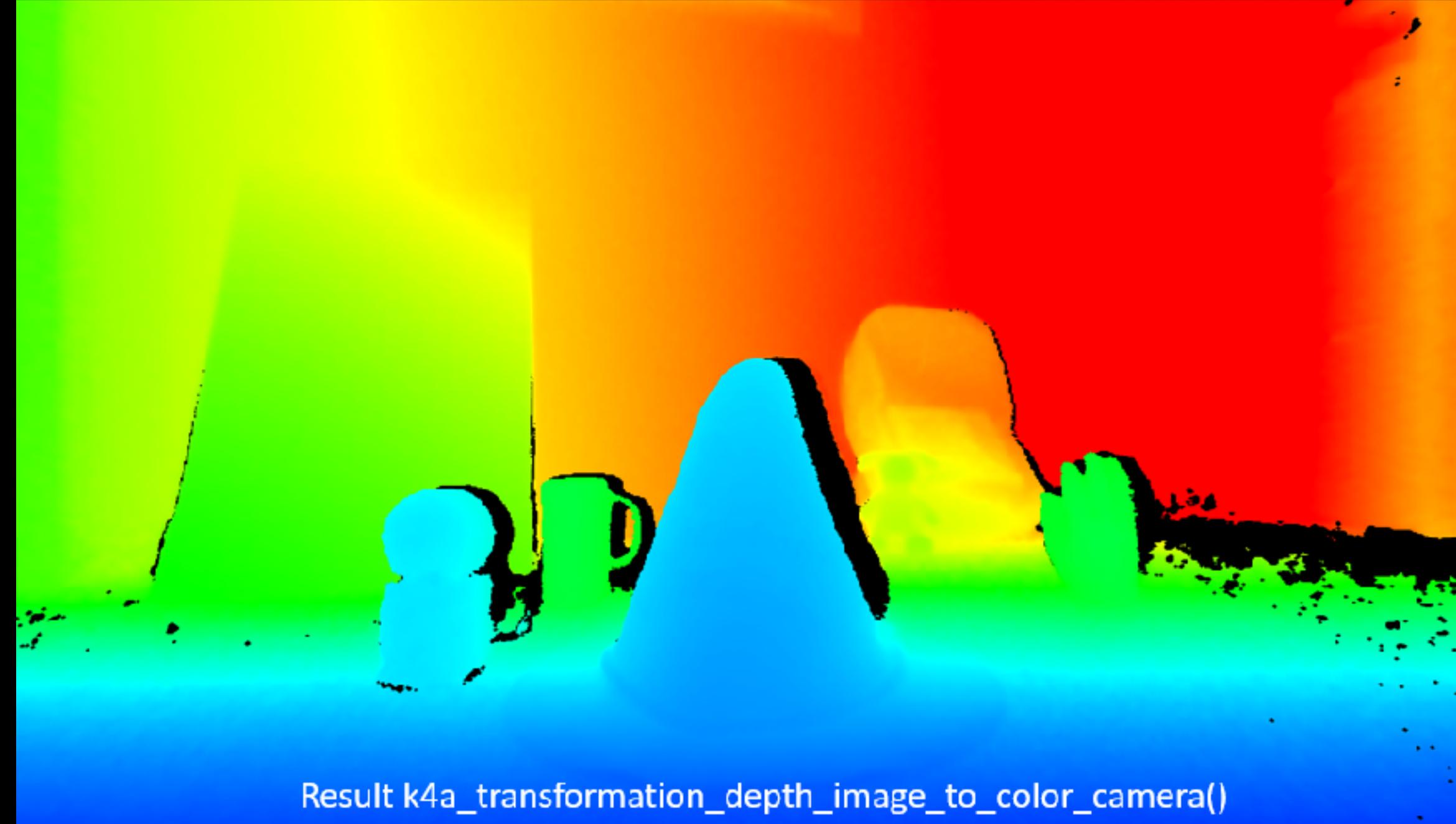
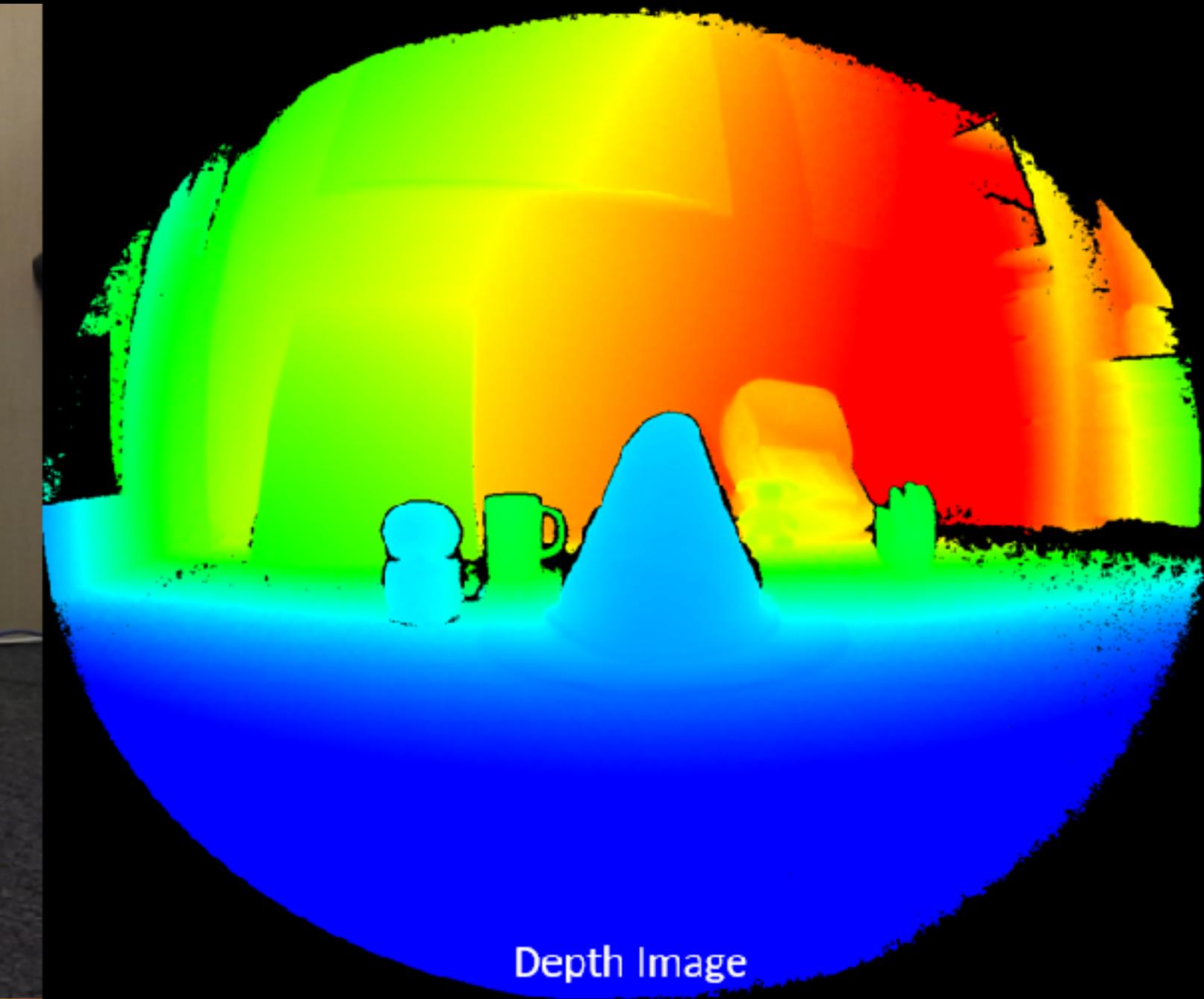
motion sensing devices

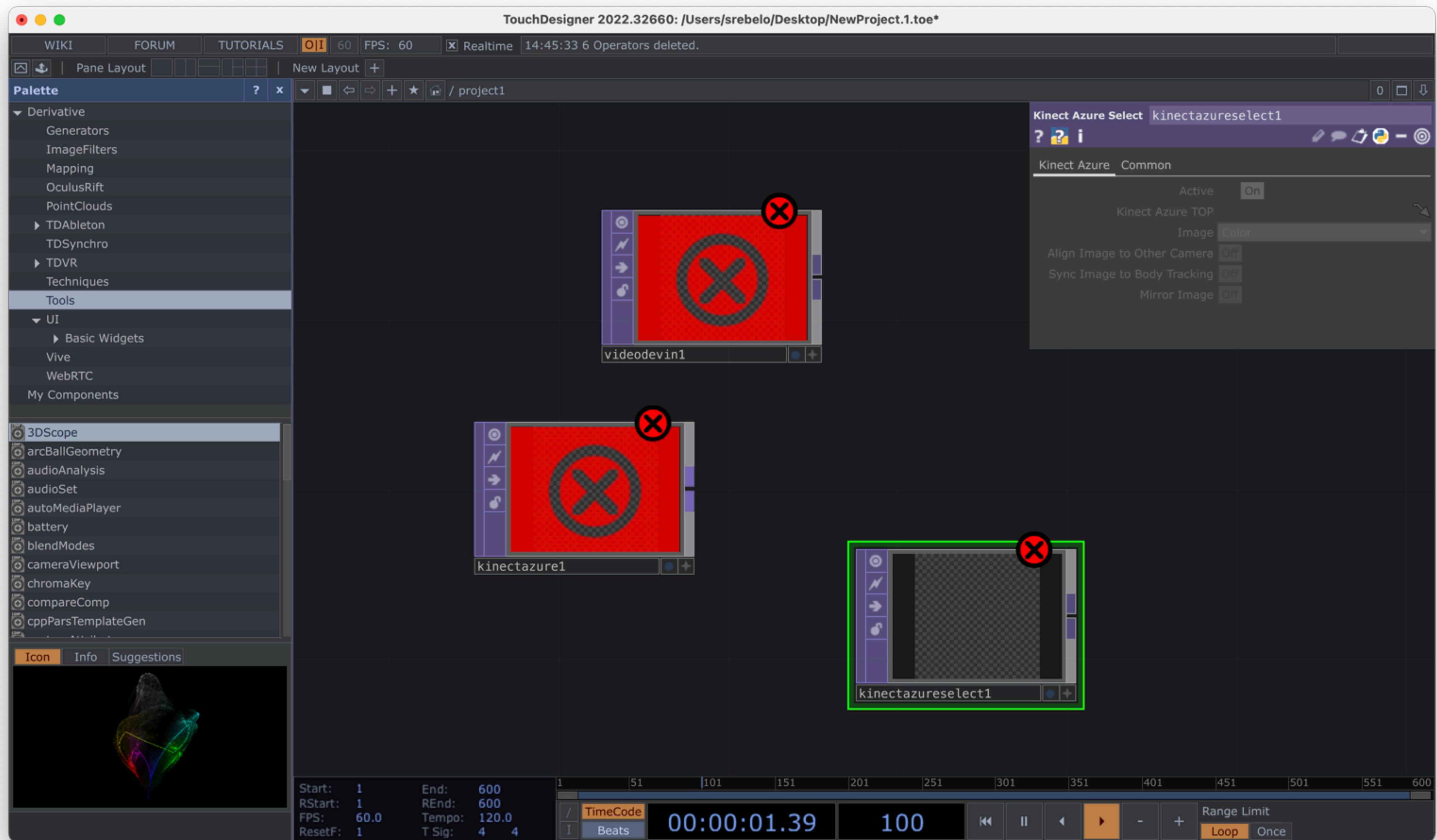


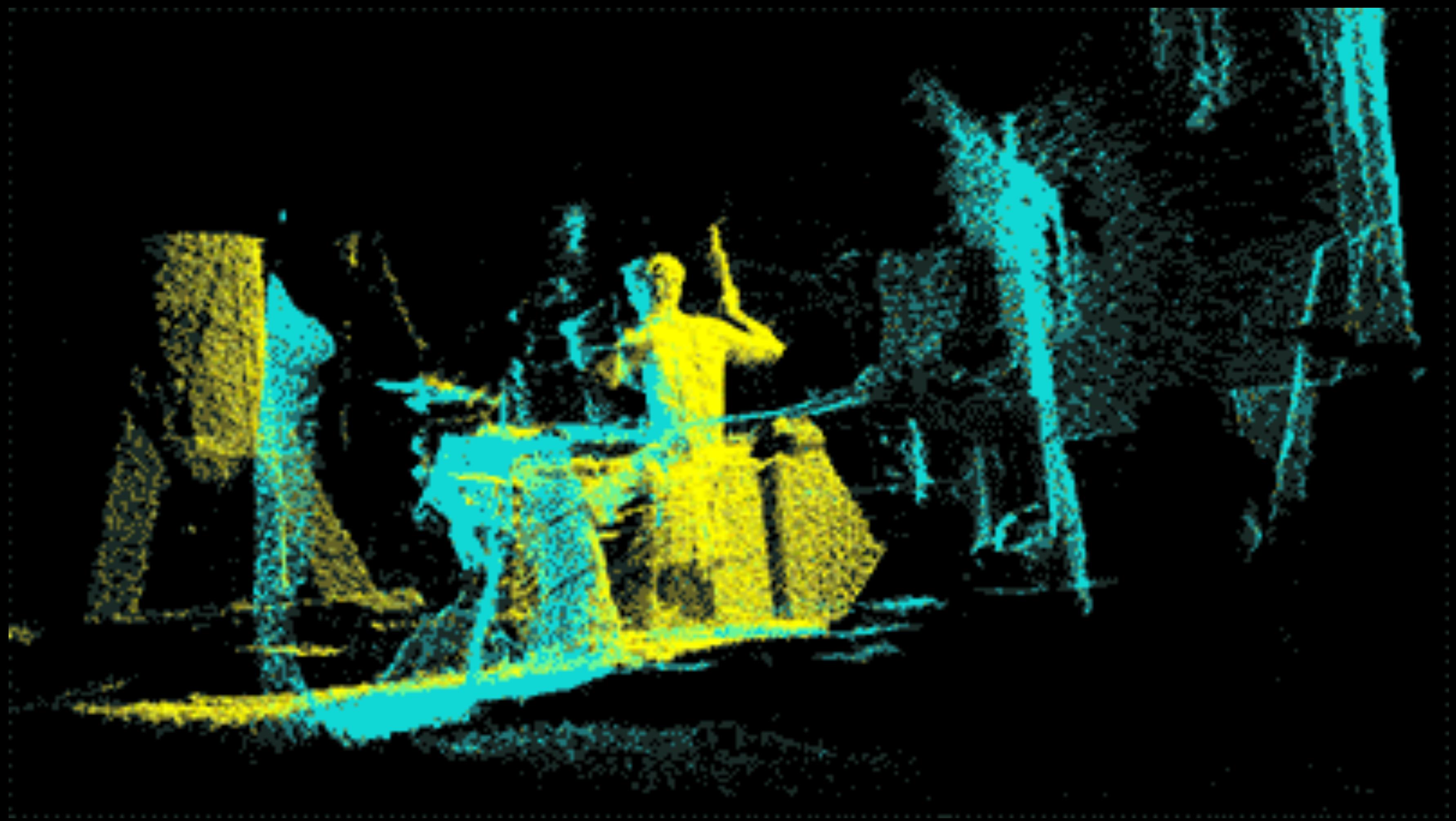
Kinect 2 for Xbox 360 (2014)

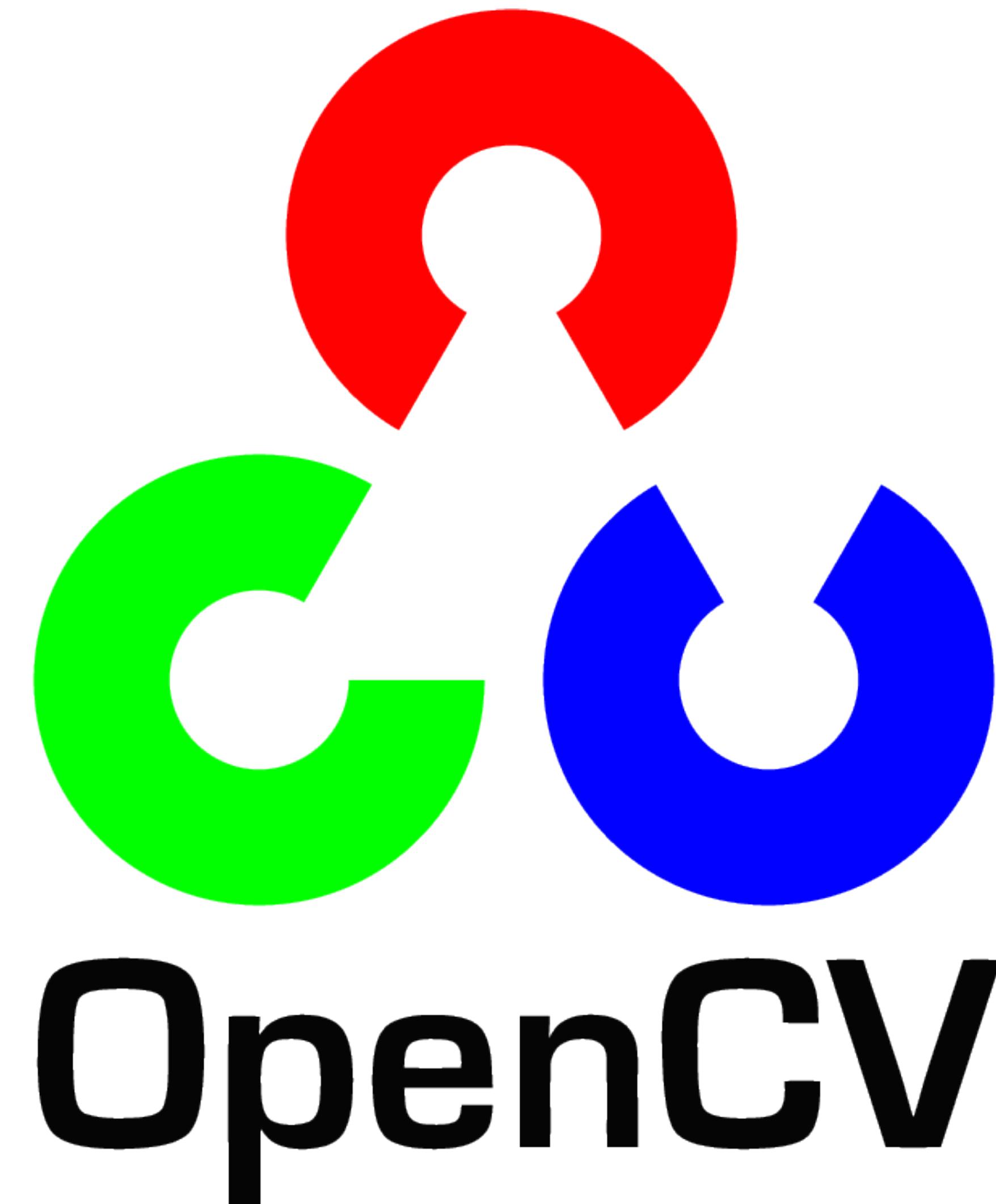


Azure Kinect DK (2020)









<https://github.com/opencv/opencv>

Original image



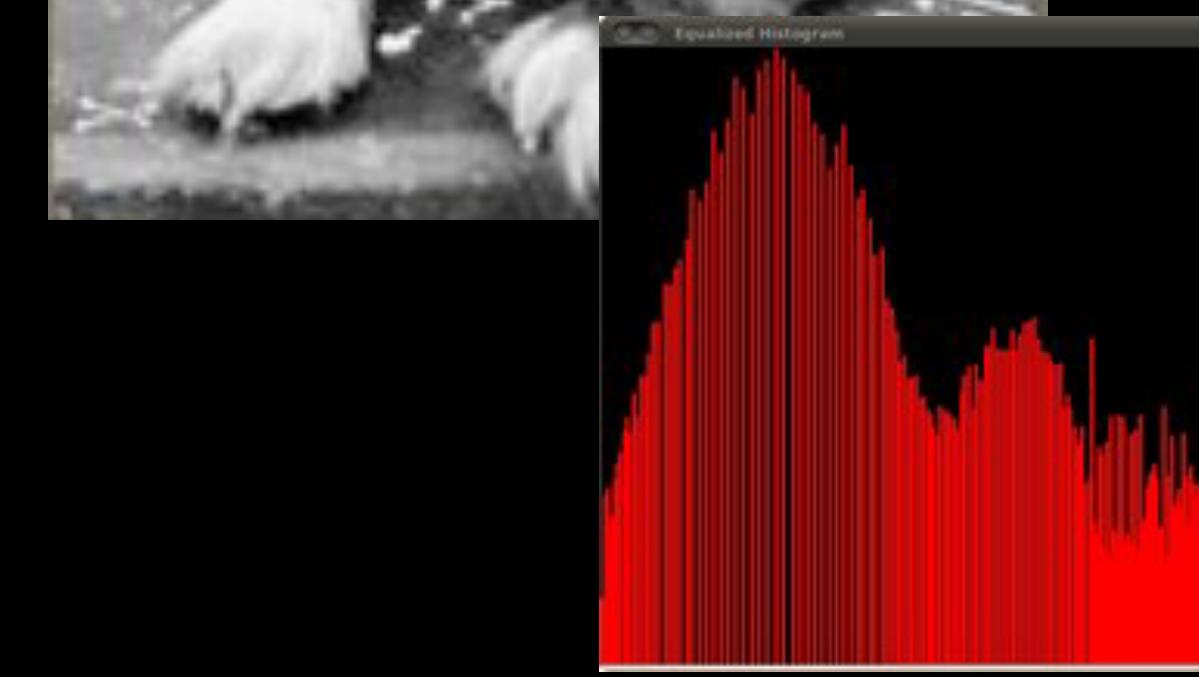
Erosion



Dilation



Image processing filters



Color processing

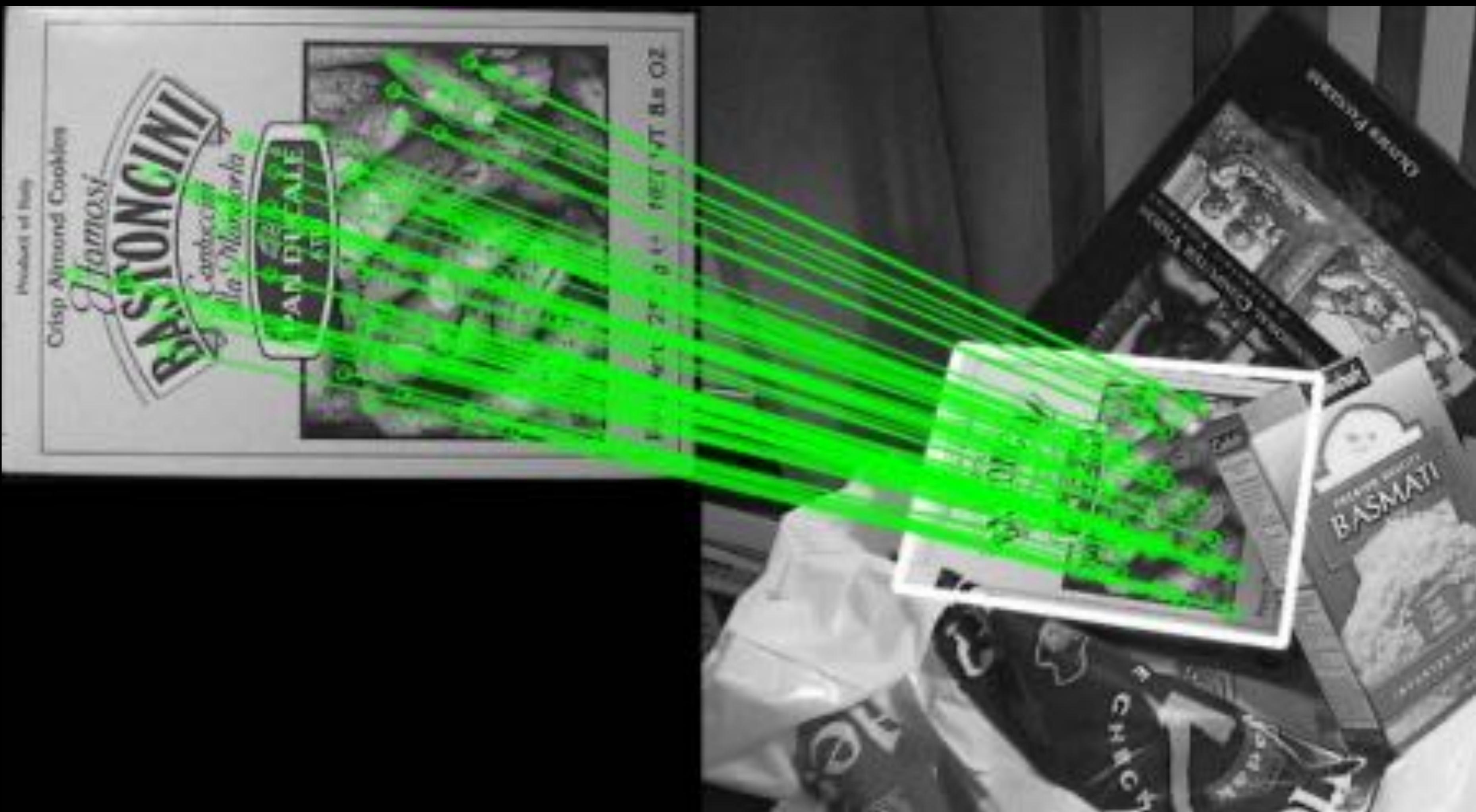
Original image



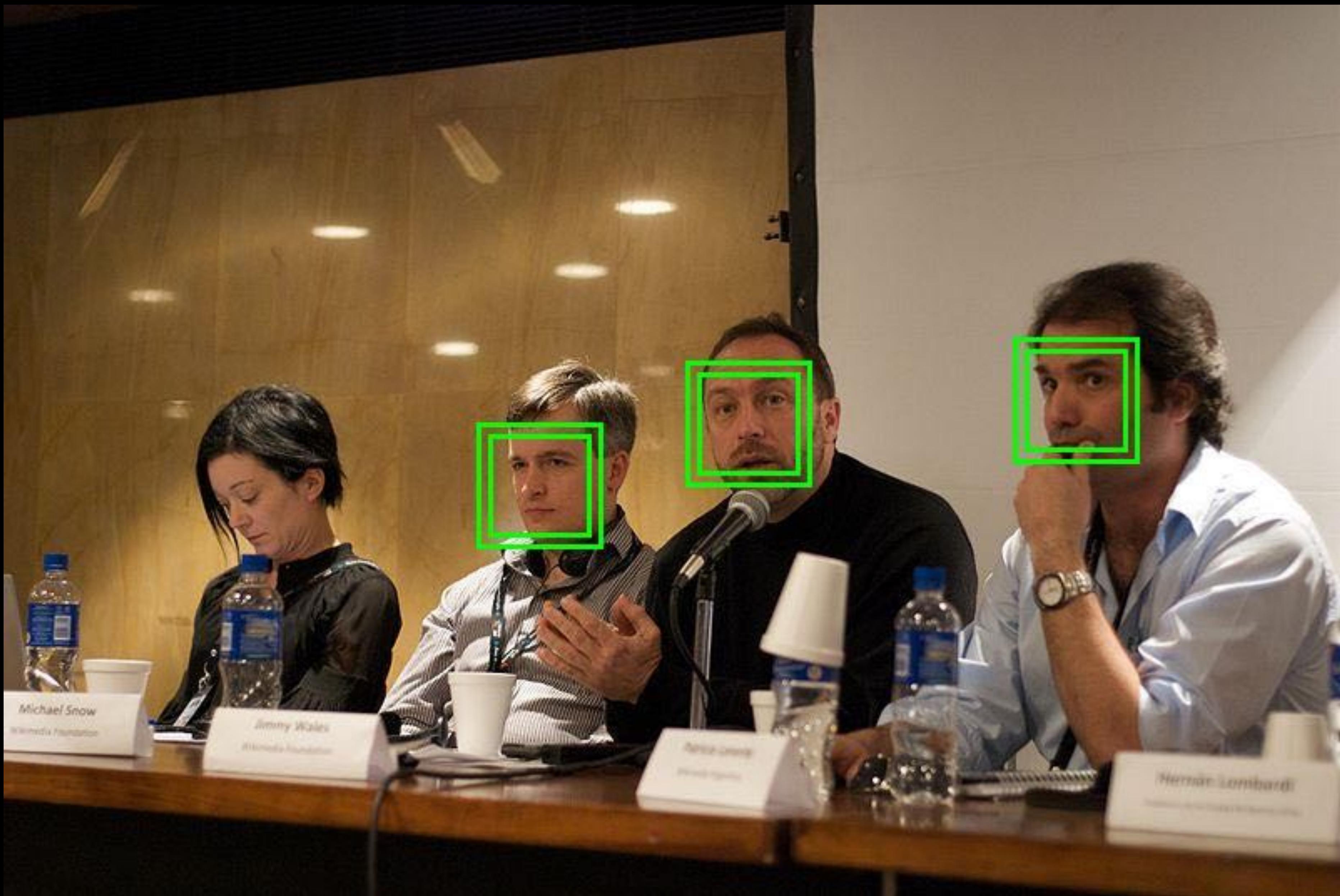
Canny Edge Image



Edge detection filters



Feature-extraction for homography



Pattern matching with machine-learning, using cascading algorithms.

The screenshot shows a GitHub repository page for the user `atduskgreg` with the repository name `opencv-processing`. The repository is public and has 45 issues, 1 pull request, and 155 commits. It includes sections for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and Insights. The Code section is currently selected. The repository description is "OpenCV for Processing. A creative coding computer vision library based on the official OpenCV Java API". It features a sidebar with links for Readme, View license, 1.3k stars, 115 watching, 471 forks, and Report repository. There are 14 releases, with the latest being 0.5.4. The Packages section is also visible.

<https://github.com/atduskgreg/opencv-processing>

atduskgreg / opencv-processing Public

Code Issues 45 Pull requests 1 Actions Projects Wiki Security Insights

master 3 branches 15 tags Go to file Code

atduskgreg merge in a couple of extra pull requests from devin e1fc182 on May 22, 2017 155 commits

data Updating for working export on mac os, working on widows, and e... 10 years ago

examples merge in a couple of extra pull requests from devin 6 years ago

lib Fix rpath on linux-armv6hf files 8 years ago

resources merge in a couple of extra pull requests from devin 6 years ago

src/gab/opencv Merge pull request #77 from qwzybug/morphological-closing 6 years ago

web initial commit 10 years ago

.classpath Updated loading of native lib for the new processing 2.1 export m... 10 years ago

.gitignore add libs back to gitignore, remove local paths from classpath 9 years ago

.project Replace instances of OpenCVPro in readme. 10 years ago

license.txt initial commit 10 years ago

readme.md Support for morphological operations like opening/closing et al. 8 years ago

About

OpenCV for Processing. A creative coding computer vision library based on the official OpenCV Java API

Readme View license 1.3k stars 115 watching 471 forks Report repository

Releases 14

0.5.4 Latest on May 22, 2017 + 13 releases

Packages

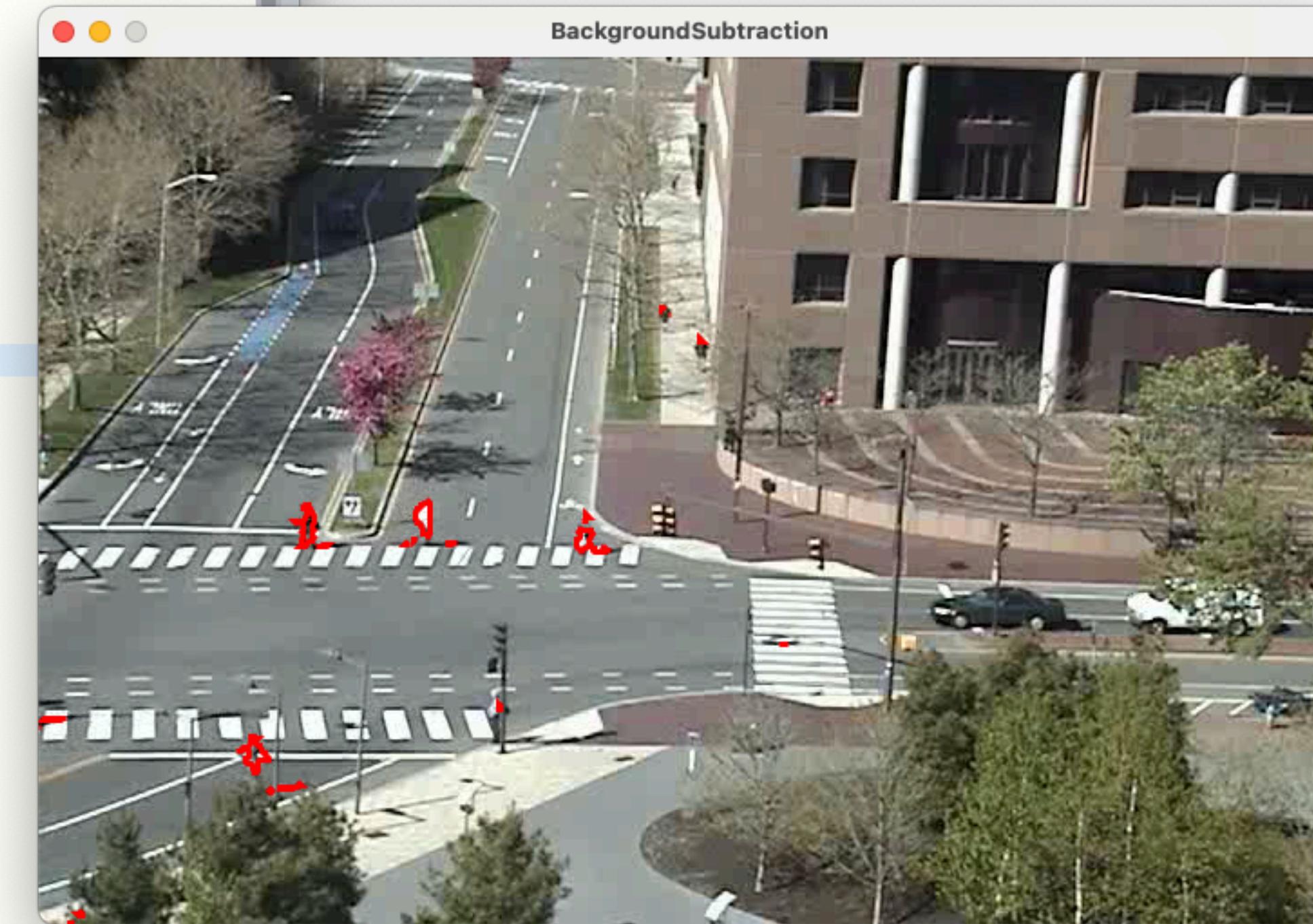
BackgroundSubtraction | Processing 4.2

BackgroundSubtraction

```
1 import gab.opencv.*;
2 import processing.video.*;
3
4 Movie video;
5 OpenCV opencv;
6
7 void setup() {
8     size(720, 480);
9     video = new Movie(this, "street.mov");
10    opencv = new OpenCV(this, 720, 480);
11
12    opencv.startBackgroundSubtraction(5, 3, 0.5);
13
14    video.loop();
15    video.play();
16 }
17
18 void draw() {
19     image(video, 0, 0);
20
21     if (video.width == 0 || video.height == 0)
22         return;
23
24     opencv.loadImage(video);
25     opencv.updateBackground();
26
27     opencv.dilate();
28     opencv.erode();
29
30     noFill();
31     stroke(255, 0, 0);
32     strokeWeight(3);
33     
```

Processing video library using bundled GStreamer 1.20.3
Scanning GStreamer plugins... Done.
Java OpenCV 4.6.0

Console Errors



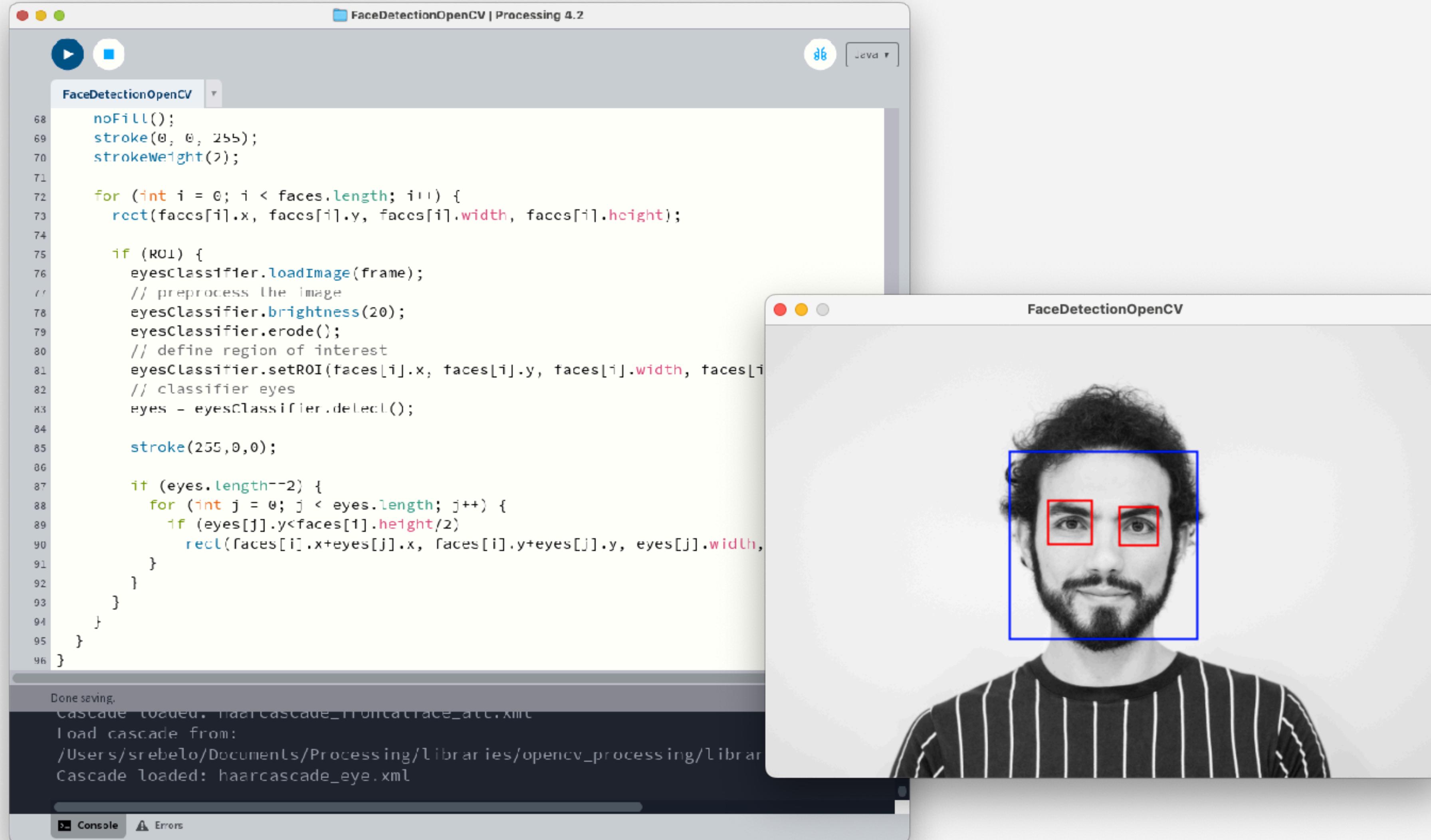
FaceDetectionOpenCV | Processing 4.2

FaceDetectionOpenCV

```
68 noFill();
69 stroke(0, 0, 255);
70 strokeWeight(2);
71
72 for (int i = 0; i < faces.length; i++) {
73     rect(faces[i].x, faces[i].y, faces[i].width, faces[i].height);
74
75     if (ROI) {
76         eyesClassifier.loadImage(frame);
77         // preprocess the image
78         eyesClassifier.brightness(20);
79         eyesClassifier.erode();
80         // define region of interest
81         eyesClassifier.setROI(faces[i].x, faces[i].y, faces[i].width, faces[i].height);
82         // classifier eyes
83         eyes = eyesClassifier.detect();
84
85         stroke(255,0,0);
86
87         if (eyes.length>2) {
88             for (int j = 0; j < eyes.length; j++) {
89                 if (eyes[j].y<faces[1].height/2)
90                     rect(faces[i].x+eyes[j].x, faces[i].y+eyes[j].y, eyes[j].width,
91                         eyes[j].height);
92             }
93         }
94     }
95 }
96 }
```

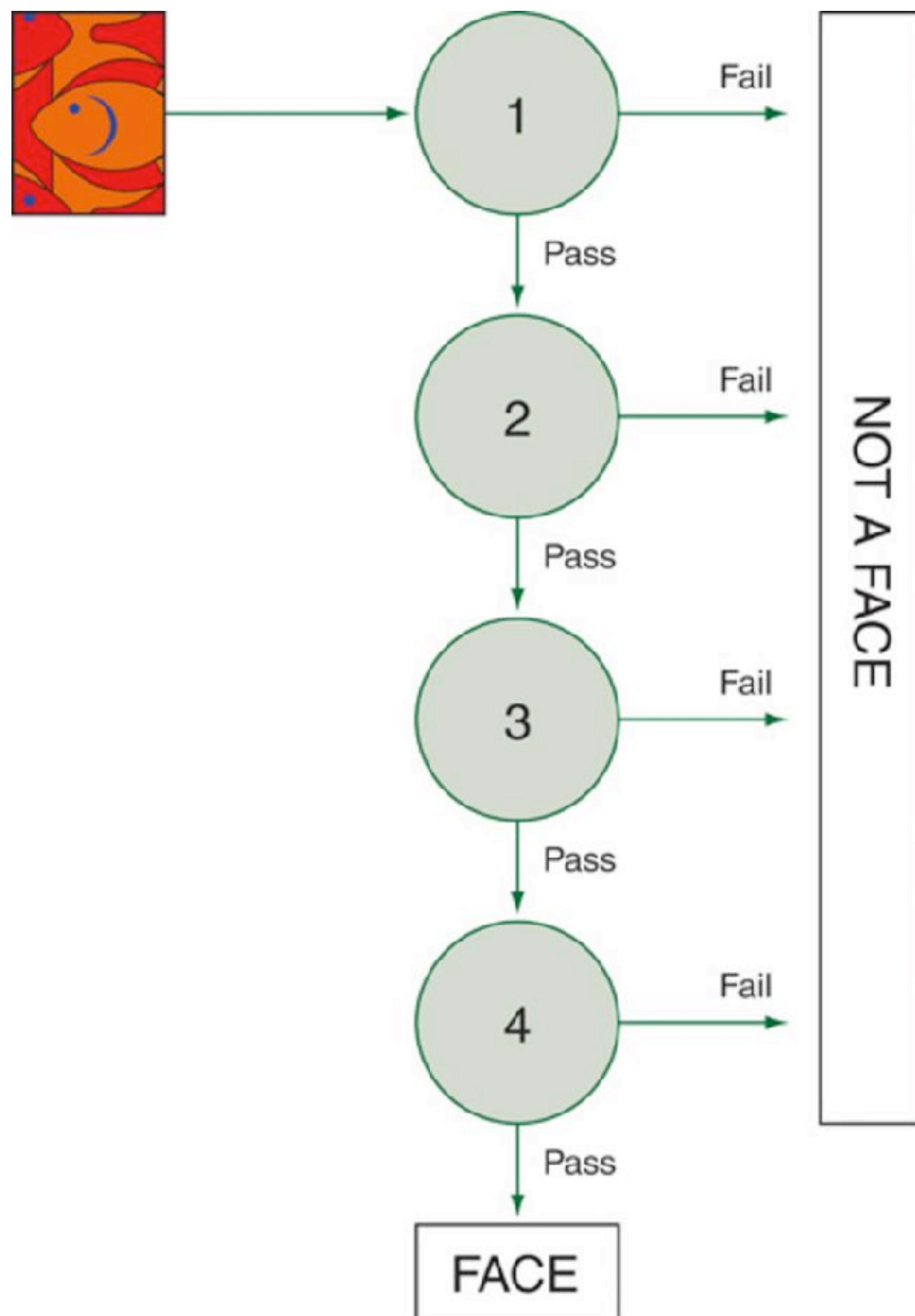
Done saving.
Cascade loaded: haarcascade_frontalface_alt.xml
Load cascade from:
/Users/srebelo/Documents/Processing/libraries/opencv_processing/library
Cascade loaded: haarcascade_eye.xml

Console Errors

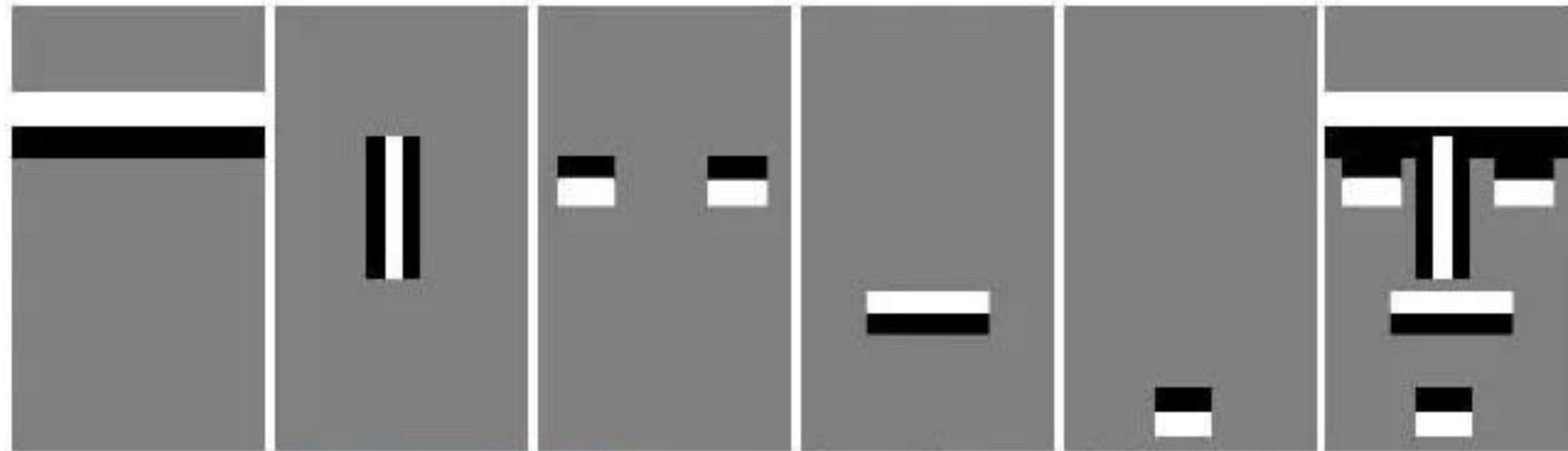


Face Detection

<https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/4-FaceDetection/FaceDetectionOpenCV>



Viola-jones Detector



A comparative study between LBP and Haar-like features for Face Detection using OpenCV. K. Kadir et al. (2014)

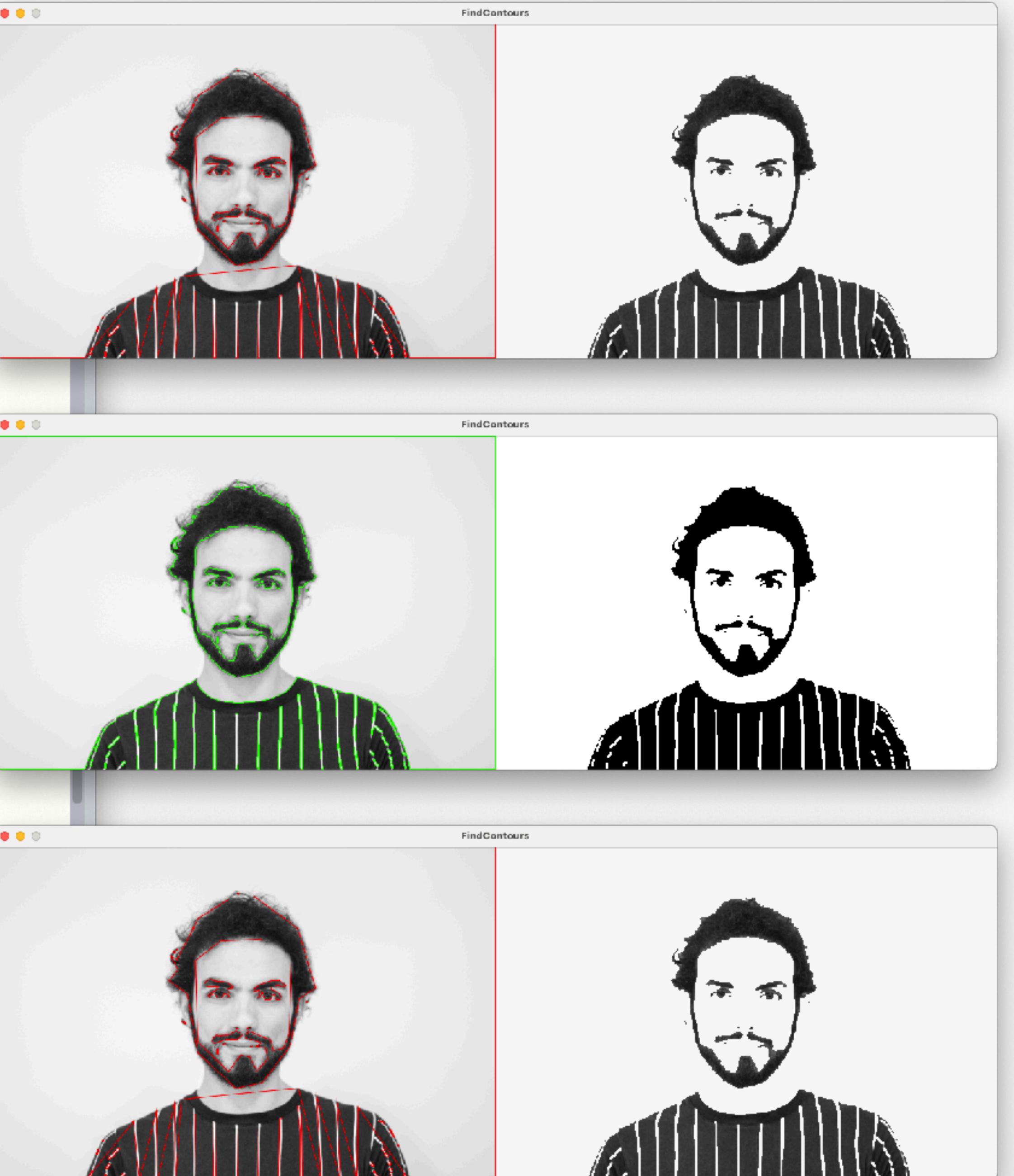
FindContours | Processing 4.2

```

63 contours = opencv.findContours();
64 // println("found", contours.size(), "contours");
65
66 image(frame, 0, 0);
67
68 if (mask) {
69   result.filter(INVERT);
70   // ensure that result has the same size than frame
71   result = result.get(0,0, frame.width, frame.height);
72   PImage copiedFrame = frame.copy();
73   copiedFrame.mask(result);
74   image(copiedFrame, frame.width, 0);
75 } else {
76   image(result, frame.width, 0);
77 }
78
79 noFill();
80 strokeWeight(1);
81
82 for (Contour contour : contours) {
83   color c = polygonApproximation ? color(255, 0, 0) : color(0, 255, 0);
84   // draw contour
85   if (!polygonApproximation) {
86     stroke(c);
87     contour.draw();
88   } else {
89     stroke(c);
90     beginShape();
91     for (PVector point : contour.getPolygonApproximation().getPoints()) {
92       vertex(point.x, point.y);
93     }
94     endShape();
95   }
96 }
97
98 }
99 
```

Scanning GStreamer plugins... Done.
[0] "FaceTime HD Camera"
[1] "Sérgio Rebelo Camera"
Java OpenCV 4.6.0

Console Errors



Contour Tracing Algorithms

<https://github.com/sengiomrebelo/amazing-robots-cv/tree/main/Demos/5-FindContours>

github.com

Deep Vision Processing

Build passing

Deep computer-vision algorithms for [Processing](#).

The idea behind this library is to provide a simple way to use (inference) machine learning algorithms for computer vision tasks inside Processing. Mainly portability and easy-to-use are the primary goals of this library. Starting with version `0.6.0` CUDA inferencing support is built into the library (Windows & Linux).

Caution: The API is still in development and can change at any time.

Languages

Java 100.0%

Build passing

FaceDetectorExample | Processing 4.2

FaceDetectorExample

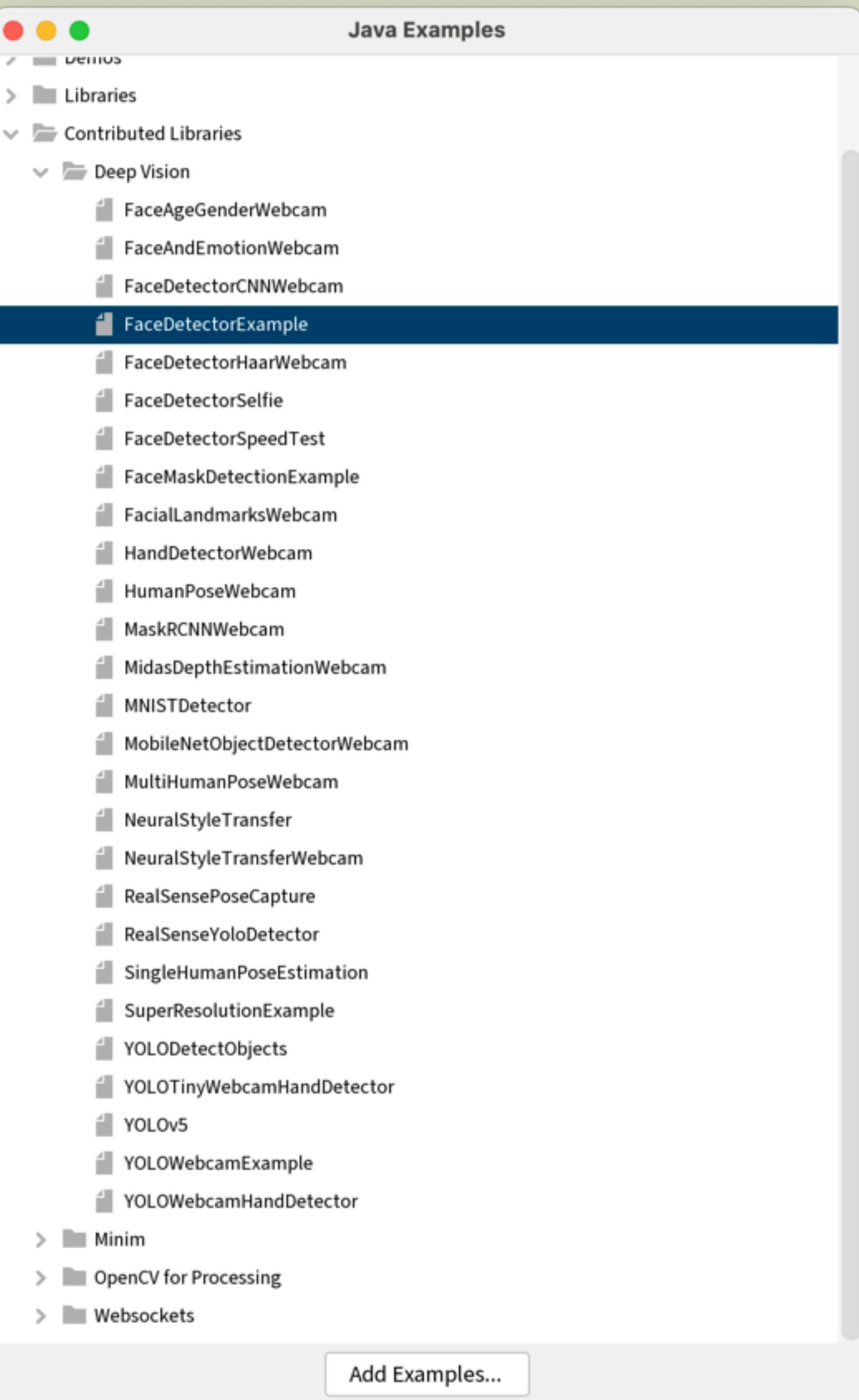
```
37
38 detections = network.run(testImage);
39 image(testImage, 0, 0);
40
41 noFill();
42 strokeWeight(2f);
43
44 stroke(200, 80, 100);
45 for (ObjectDetectionResult detection : detections) {
46   rect(detection.getX(), detection.getY(), detection.getW()
47 }
48
49 surface.setTitle("Face Recognition Test - FPS: " + M
50 }
```

face [0.9847959]
face [0.9781659]
face [0.72960687]
found 4 faces!

Console Errors

YOLO Test - FPS: 60

The image shows a dual-pane interface of a Processing application. The left pane displays the source code for 'FaceDetectorExample', which includes imports for PImage and ObjectDetectionResult, and logic for running a neural network on a test image and drawing the results. The right pane shows a live video feed from a camera or file, with numerous bounding boxes of different colors (yellow, red, orange) drawn over the scene to identify objects. Labels such as 'traffic light', 'bus', 'car', 'motorbike', 'person', and 'bicycle' are placed near their respective detections. The title bar of the right pane indicates a frame rate of 'FPS: 60'. The overall environment is a Mac OS X desktop.



FaceDetectorExample | Processing 4.2

FaceDetectorExample

```
37
38 detections = network.run(testImage);
39 image(testImage, 0, 0);

40 noFill();
41 strokeWeight(2f);

42 stroke(200, 80, 100);
43 for (ObjectDetectionResult detection : detections) {
44   rect(detection.getX(), detection.getY(), detection.get
45 }
46
47 surface.setTitle("Face Recognition Test - FPS: " + Math
48
49
50 }
```

face [0.9847959]
face [0.9781659]
face [0.72960687]
found 4 faces!

Java

Face Recognition Test - FPS: 60

Console Errors

Face Detection

Examples → Contributed Libraries → DeepVision → FaceDetectorExample

or <https://github.com/sengiomrebelo/amazing-robots-ov/tree/main/Demos/4-FaceDetection/FaceDetectorML>

YOLOv5 | Processing 4.2

YOLOv5

```
1 /**
2 * Tecnologias de Interface, Winter 2023
3 * Universidade de Coimbra
4 * MSc in Design and Multimedia
5 *
6 * Week 9
7 * Object Detection (YOLO)
8 *
9 * @authors: Florian Bruggisser, Sérgio M. Rebelo, Ana Cláudia
10 * @since: 04-05-2023
11 * @based: based FaceDetector Example from DeepVision
12 */
13
14 import ch.bildspur.vision.*;
15 import ch.bildspur.vision.result.*;
16
17 DeepVision deepVision = new DeepVision(this);
18 YOLONetwork yolo;
19 ResultList<ObjectDetectionResult> detections;
20
21 String [] paths = {"image-1.jpg", "image-2.jpg", "image-3.jpg",
22 int current = 0;
23
24 PImage image;
25 int textSize = 12;
26
27 public void setup() {
28   size(640, 480);
29
30   colorMode(HSB, 360, 100, 100);
31
32   config();
33 }
```

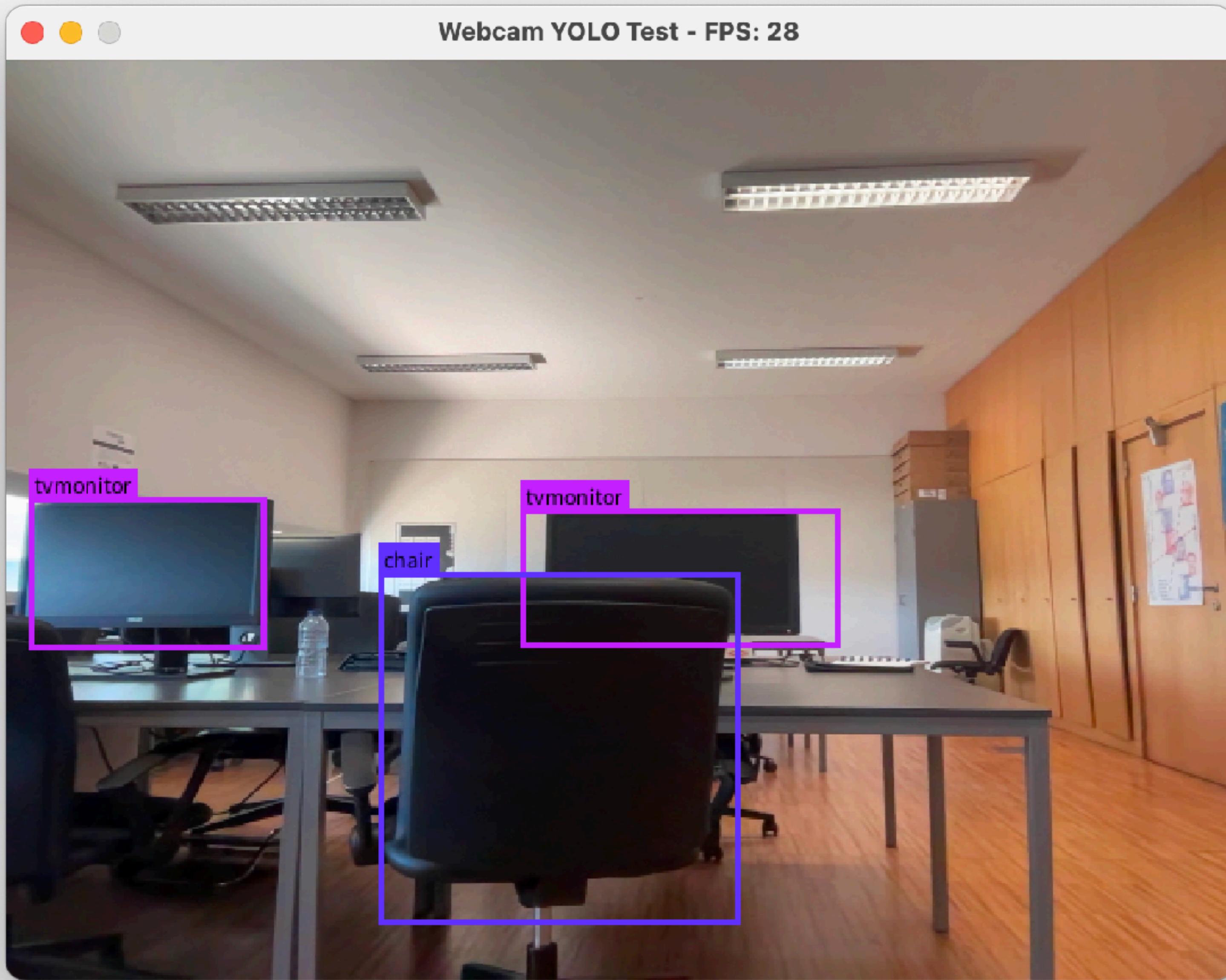
DNN OpenCL backend enabled
DNN OpenCL backend enabled
DNN OpenCL backend enabled
DNN OpenCL backend enabled

YOLO Test - FPS: 60

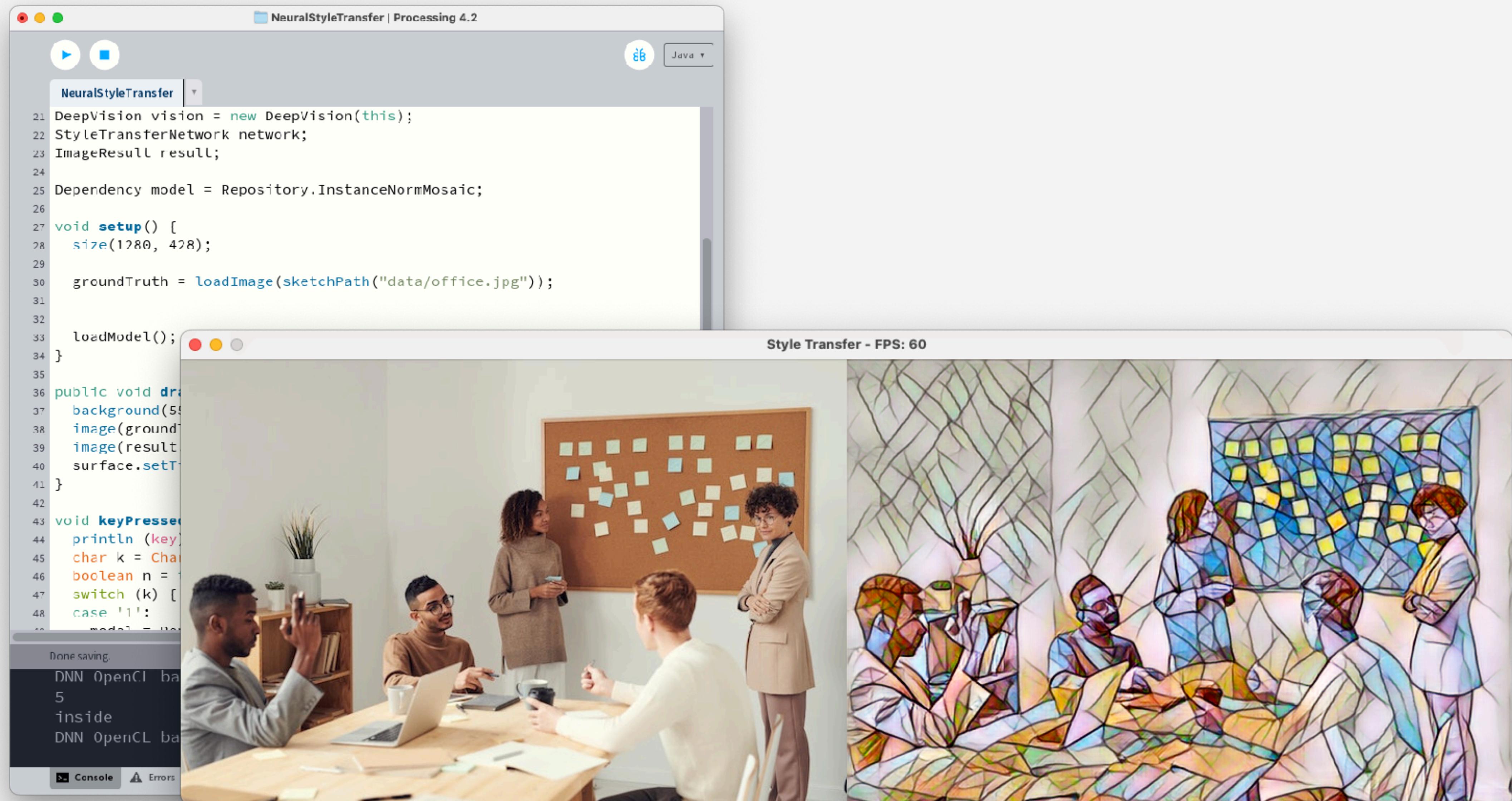


Object Detection (YOLO)

<https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/6-ObjectDetection/YOLO>



<https://github.com/sergiomrebelo/amazing-robots-cv/tree/main/Demos/6-ObjectDetection/webcamYOLO>

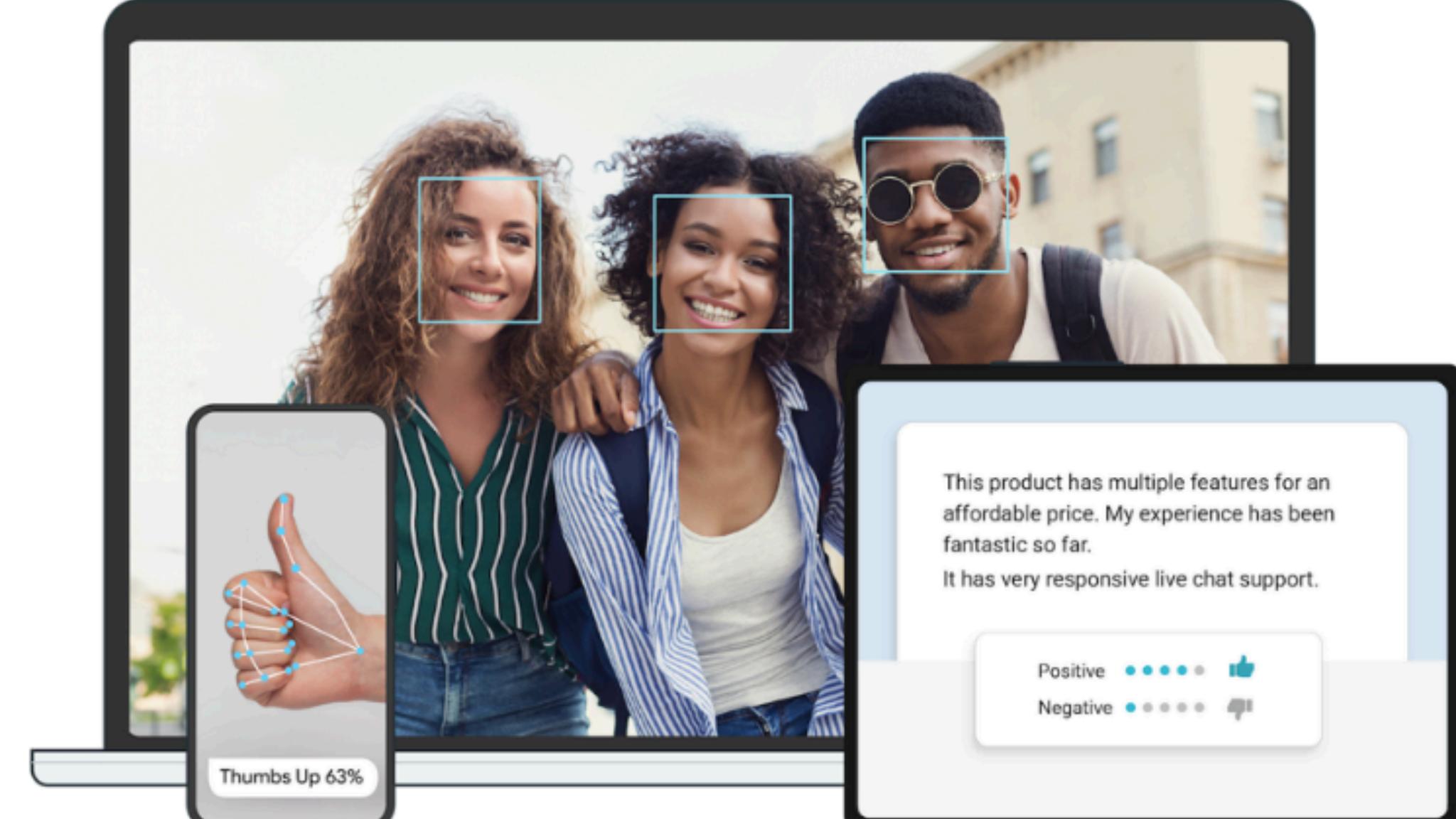


Attention: This MediaPipe Solutions Preview is an early release. [Learn more.](#)

On-device machine learning for everyone

Delight your customers with innovative machine learning features. MediaPipe contains everything that you need to customize and deploy to mobile (Android, iOS), web, desktop, edge devices, and IoT, effortlessly.

[See demos](#) [Learn more](#)



Easy to use

Self-serve ML solutions with simple-to-use abstractions. Use low-code APIs or no-code studio to customize, evaluate, prototype, and deploy.

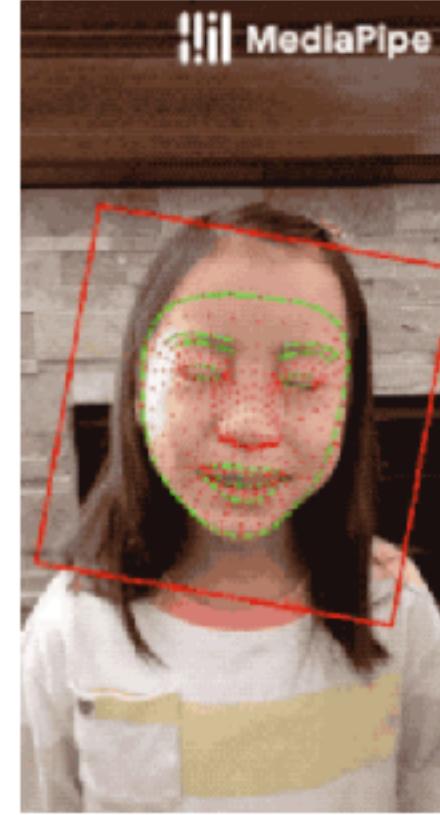
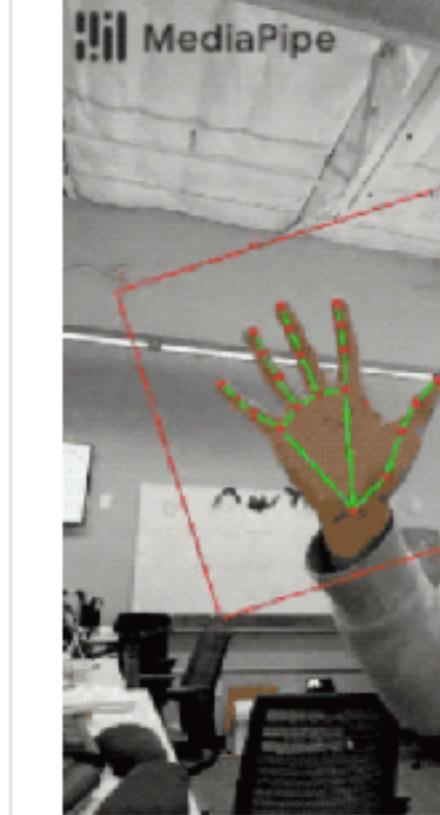
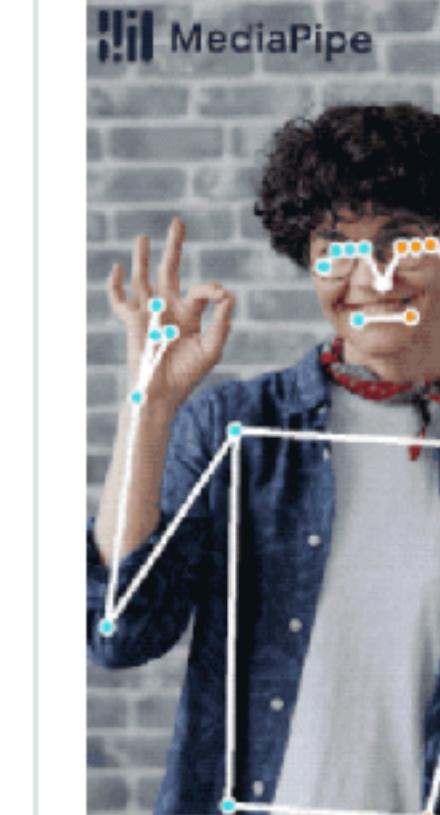
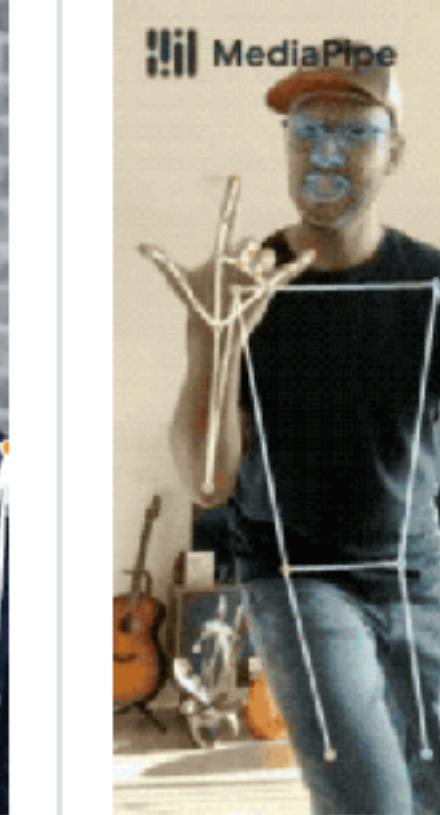
Innovative

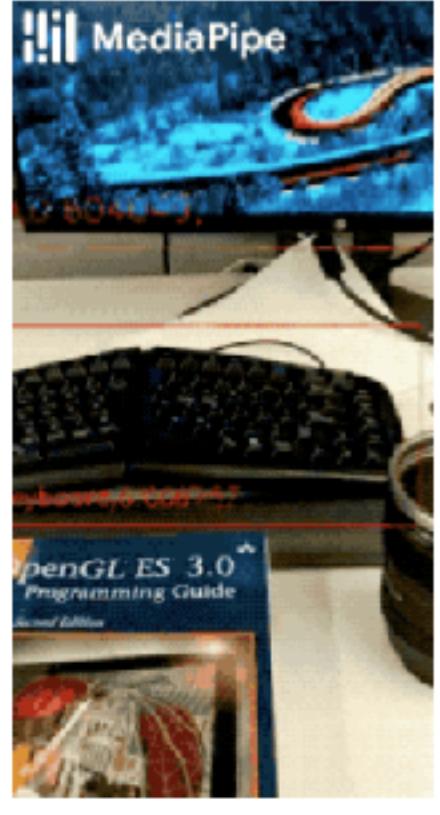
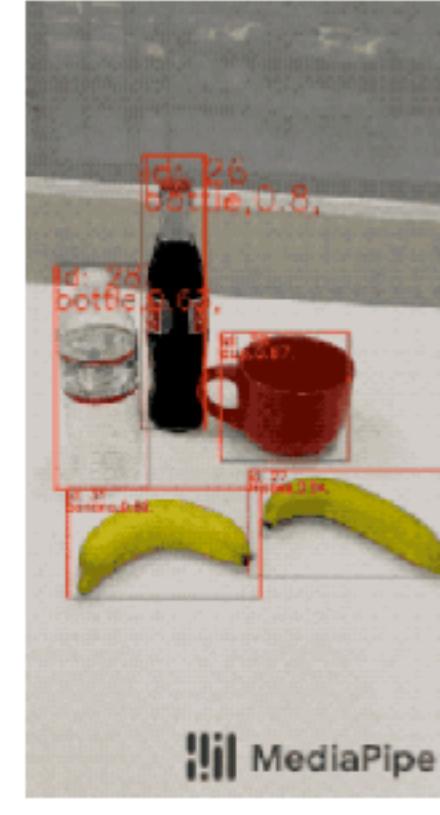
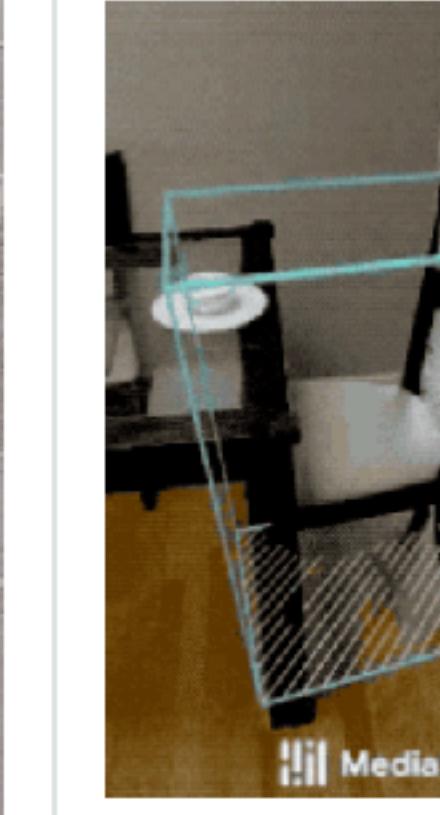
Advanced ML solutions for popular tasks, crafted with Google ML expertise.

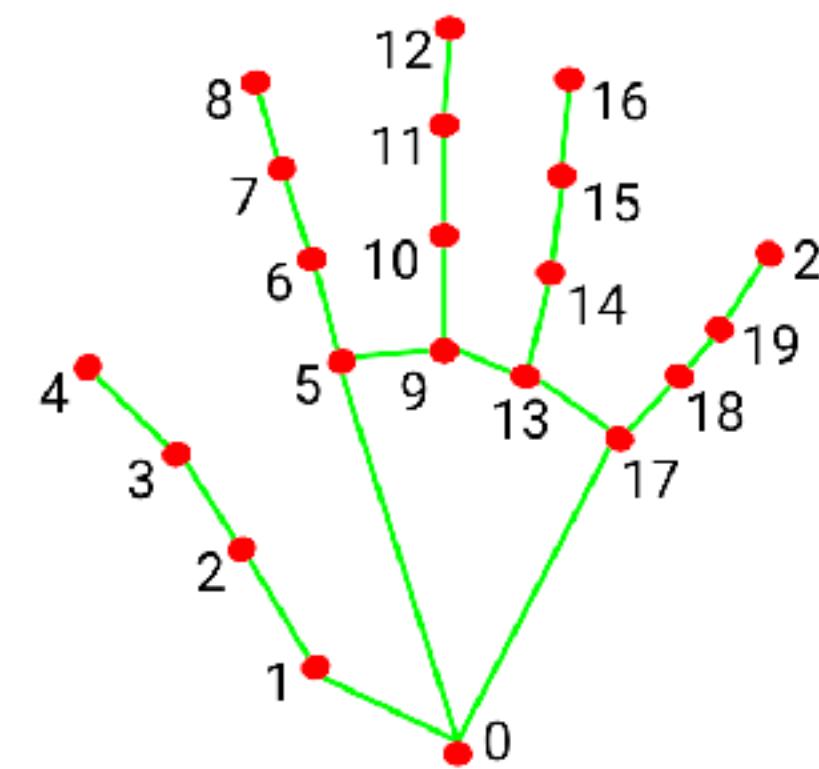
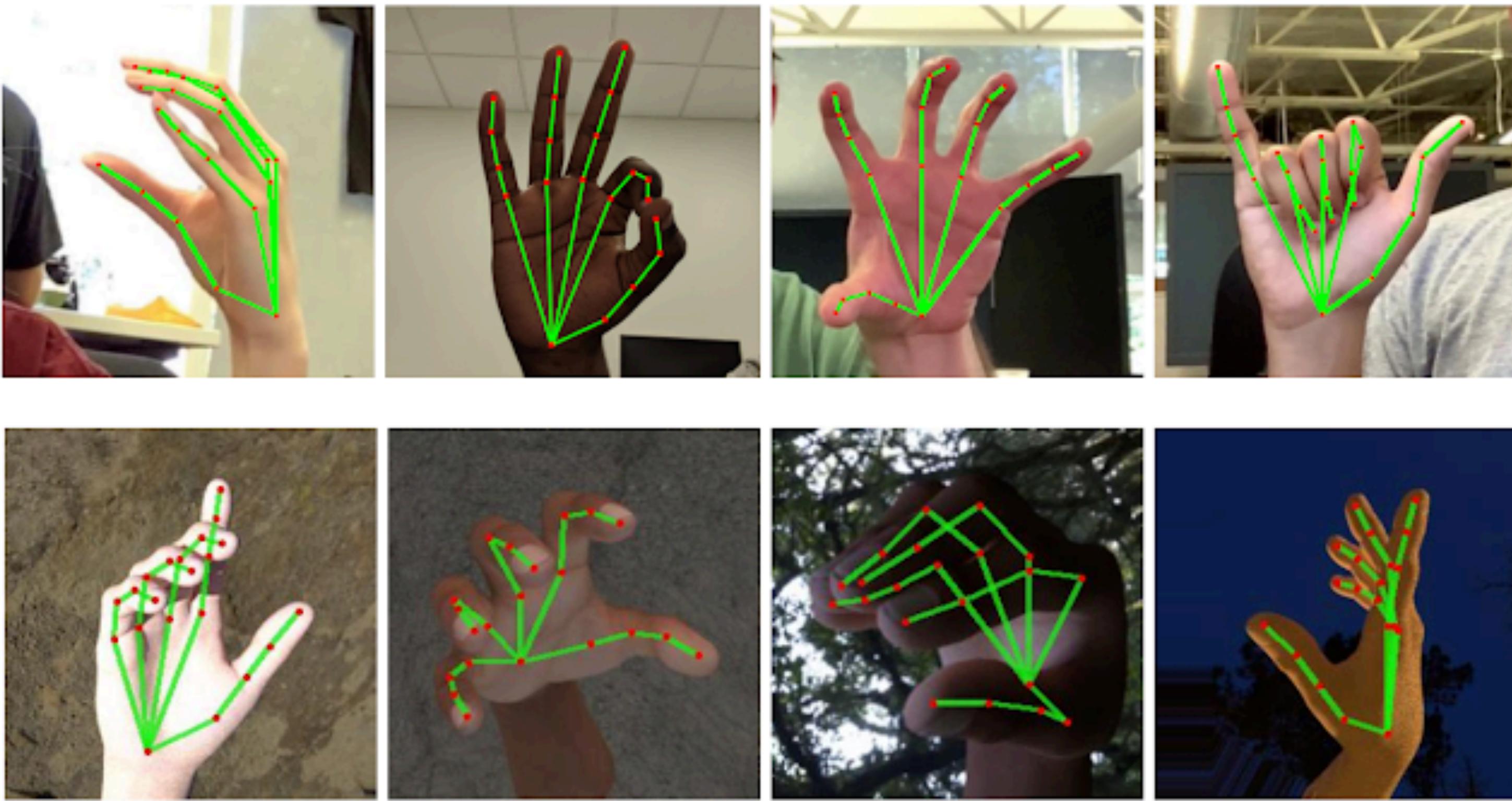
Fast, really fast

End-to-end optimization, including hardware acceleration, all while lightweight enough to run well on battery-powered devices.

ML solutions in MediaPipe

Face Detection	Face Mesh	Iris	Hands	Pose	Holistic
					

Hair Segmentation	Object Detection	Box Tracking	Instant Motion Tracking	Objectron	KNIFT
					



- | | |
|-----------------------|-----------------------|
| 0. WRIST | 11. MIDDLE_FINGER_DIP |
| 1. THUMB_CMC | 12. MIDDLE_FINGER_TIP |
| 2. THUMB_MCP | 13. RING_FINGER_MCP |
| 3. THUMB_IP | 14. RING_FINGER_PIP |
| 4. THUMB_TIP | 15. RING_FINGER_DIP |
| 5. INDEX_FINGER_MCP | 16. RING_FINGER_TIP |
| 6. INDEX_FINGER_PIP | 17. PINKY_MCP |
| 7. INDEX_FINGER_DIP | 18. PINKY_PIP |
| 8. INDEX_FINGER_TIP | 19. PINKY_DIP |
| 9. MIDDLE_FINGER_MCP | 20. PINKY_TIP |
| 10. MIDDLE_FINGER_PIP | |

The screenshot shows a browser window with the URL <https://justadudewhohacks.github.io/face-api.js/docs/index.html>. The page title is "face-api.js". The main content area displays the "face-api.js" documentation, which includes a brief introduction, a table of contents, and a sidebar of global symbols.

face-api.js

JavaScript API for face detection and face recognition in the browser implemented on top of the tensorflow.js core API ([tensorflow/tfjs-core](#))

Table of Contents:

- [Resources](#)
 - [Live Demos](#)
 - [Tutorials](#)
- [Examples](#)
 - [Running the Examples](#)
- [Available Models](#)
 - [Face Detection Models](#)
 - [68 Point Face Landmark Detection Models](#)
 - [Face Recognition Model](#)
 - [Face Expression Recognition Model](#)
- [Getting Started](#)
 - [face-api.js for the Browser](#)
 - [face-api.js for Nodejs](#)
- [Usage](#)
 - [Loading the Models](#)
 - [High Level API](#)
 - [Displaying Detection Results](#)

Globals

- [TinyYolov2SizeType](#)
- [BoundingBox](#)
- [Box](#)
- [BoxWithText](#)
- [ComposableTask](#)
- [ComputeAllFaceDescriptorsTask](#)
- [ComputeFaceDescriptorsTaskBase](#)
- [ComputeSingleFaceDescriptorTask](#)
- [DetectAllFaceLandmarksTask](#)
- [DetectAllFacesTask](#)
- [DetectFaceLandmarksTaskBase](#)
- [DetectFacesTaskBase](#)
- [DetectSingleFaceLandmarksTask](#)
- [DetectSingleFaceTask](#)
- [Dimensions](#)
- [FaceDetection](#)
- [FaceDetectionNet](#)
- [FaceExpressionNet](#)
- [FaceFeatureExtractor](#)
- [FaceLandmark68Net](#)
- [FaceLandmark68TinyNet](#)
- [FaceLandmarkNet](#)
- [FaceLandmarks](#)
- [FaceLandmarks5](#)
- [FaceLandmarks68](#)
- [FaceMatch](#)
- [FaceMatcher](#)

The screenshot shows the homepage of the [ml5.js](https://ml5js.org) website. The page has a white background with various hand-drawn style illustrations related to machine learning and web development. At the top left is the [ml5.js](https://ml5js.org) logo. At the top right are navigation links: **Getting Started**, **Reference**, **Learn**, **Community**, and **About**. The main title **Friendly Machine Learning for the Web** is centered in a large, bold, dark font. Below it is a subtitle: **A neighborly approach to creating and exploring artificial intelligence in the browser.** In the center, there is a **Get Started** button. The page is filled with icons such as a person at a computer, a brain, puzzle pieces, a neural network, a rocket ship, a brain with thought bubbles, a lightbulb, hands, a bird, a book labeled "ml5.js", and speech bubbles.

ml5.js

Getting Started Reference Learn Community About

Friendly Machine Learning for the Web

A neighborly approach to creating and exploring artificial intelligence in the browser.

Get Started

<https://ml5js.org>

ml5.js

docs.ml5js.org

Getting Started Reference Learn Community About

Search

Welcome

Getting Started

Next Steps

FAQ

Reference

Overview

ml5 Models

BodyPose

BodySegmentation

HandPose

FaceMesh

ImageClassifier

- Description
- Quick Start
- Examples
- Step-by-Step Guide
- Properties
- Methods

SoundClassifier

Sentiment

ImageClassifier

The illustration shows a neural network architecture. On the left, a stack of horizontal grey bars represents input data. In the center, a rectangular frame contains a landscape scene with mountains and a sun. A magnifying glass is positioned over the frame, focusing on the scene. To the right, a grid of circles represents the output or hidden layers of the network.

Image Credit: Naveen | Contribute ❤️

Description

The ml5.js imageClassifier is a pre-trained model that can recognize the content of an image. It can identify objects, animals, and even people in a picture. The image classifier uses a neural network to analyze the image and provide a list of possible labels for the content of the image in its entirety.

The ml5.js imageClassifier uses the pre-trained MobileNet model by default. You can optionally load and use other models such as Darknet as well as a custom-trained model DoodleNet which is also built.

docs.ml5js.org

Getting Started Reference Learn Community About

BodyPose



Image Credit: sentya irma | Contribute ❤️

Description

The ml5.js BodyPose is a pretrained full-body pose estimation model that can estimate poses and track key body parts in real-time. It is developed leveraging TensorFlow's [MoveNet](#) and [BlazePose](#) models.

It offers flexibility for:

- Multi-person detection: Estimate poses for single or multiple people in the frame.

ml5.js

docs.ml5js.org

Getting Started Reference Learn Community About

Search

Welcome

Getting Started

Next Steps

FAQ

Reference

Overview

ml5 Models

BodyPose

BodySegmentation

- Description
- Quick Start
- Examples
- Step-by-Step Guide
- Properties
- Methods

HandPose

FaceMesh

ImageClassifier

SoundClassifier

Sentiment

Body Segmentation

The ml5.js BodySegmentation provides two models, `SelfieSegmentation` and `BodyPix`. The `SelfieSegmentation` model focuses on segmenting the human subject from the background. The `BodyPix` model is primarily used for detailed body part segmentation (e.g., distinguishing between different limbs) in images and videos. Although `BodyPix` can also perform person/background segmentation, it is more computationally intensive.

Image Credit: [ibrandify](#) | [Contribute](#) ❤️

ml5.js

docs.ml5js.org

Getting Started Reference Learn Community About

Search

Welcome

Getting Started

Next Steps

FAQ

Reference

Overview

ml5 Models

BodyPose

BodySegmentation

HandPose

- Description
- Quick Start
- Examples
- Step-by-Step Guide
- Properties
- Methods

FaceMesh

ImageClassifier

SoundClassifier

Sentiment

HandPose



A stylized illustration of two hands, one open palm-up and one slightly curled, overlaid with a light gray skeletal structure showing joint positions. The background features faint, overlapping hand silhouettes.

Image Credit: DinosoftLabs | Contribute ❤️

Description

HandPose is a machine-learning model that allows for palm detection and hand-skeleton finger tracking in the browser. It can detect multiple hands at a time and for each hand, and provides 21 2D and 3D hand keypoints that describe important locations on the palm and fingers.

The ml5.js HandPose model is based on the [HandPose implementation by TensorFlow.js](#).

docs.ml5js.org

Getting Started Reference Learn Community About

FaceMesh



Image Credit: Paweł Gieru | Contribute ❤️

Description

FaceMesh is a machine-learning model that allows for facial landmark detection in the browser. It can detect multiple faces at once and provides 468 3D facial landmarks that describe the geometry of each face. FaceMesh works best when the faces in view take up a large percentage of the image or video frame and it may struggle with small/distant faces.

The ml5.js FaceMesh model is ported from the [TensorFlow.js FaceMesh implementation](#).

ml5.js

docs.ml5js.org

SoundClassifier

Getting Started Reference Learn Community About

Search

Welcome

Getting Started

Next Steps

FAQ

Reference

Overview

ml5 Models

BodyPose

BodySegmentation

HandPose

FaceMesh

ImageClassifier

SoundClassifier

- Description
- Quick Start
- Examples
- Step-by-Step Guide
- Properties
- Methods

≡

SoundClassifier



A stylized illustration featuring a vintage-style microphone on a stand. A magnifying glass is positioned over the microphone's grille. Above the microphone is a grey cross-shaped puzzle piece. To the right of the microphone are several wavy, horizontal lines of varying shades of grey, resembling sound waves or frequency patterns.

Image Credit: Kantor Tegalsari | Contribute ❤️

Description

SoundClassifier is a machine-learning model that allows you to classify audio.

It provides the following functionalities:

- Sound identification: Detect whether a certain noise (e.g., clapping) was made or a certain word

docs.ml5js.org

Sentiment

Getting Started Reference Learn Community About

ml5.js

Overview

ml5 Models

- BodyPose
- BodySegmentation
- HandPose
- FaceMesh
- ImageClassifier
- SoundClassifier
- Sentiment
 - Description
 - Quick Start
 - Examples
 - Step-by-Step Guide
 - Properties
 - Methods

ml5 + Teachable Machine

Image + Teachable Machine

Train your own model!

NeuralNetwork

Learn

ML Classifiers

Sentiment

The illustration features two circular icons: one with a sad expression and another with a happy expression. Arrows point from the sad icon to the happy icon, suggesting a transformation or prediction process. The background includes abstract shapes like diagonal hatching and a grid of dots.

Image Credit: [kartini 1](#) | Contribute ❤️

Description

Sentiment is a model trained to predict the sentiment of any given text. For example, it can predict how positive or negative a review is with a value between 0 ("negative") and 1 ("positive").

The model is trained using IMDB reviews that have been truncated to a maximum of 200 words, only the 20000 most used words in the reviews are used.

teachablemachine.withgoogle.com

About FAQ Get Started

Teachable Machine

Train a computer to recognize your own images, sounds, & poses.

A fast, easy way to create machine learning models for your sites, apps, and more – no expertise or coding required.

Get Started

↑ ml5.js p5.js Coral nodejs TensorFlow ARDUINO

Metal 94%

Not Metal 6%

<https://www.youtube.com/watch?v=3BhkeY974Rg>

docs.ml5js.org

M Image + Teachable Machine

ml5.js

Getting Started Reference Learn Community About

Overview

ml5 Models

- BodyPose
- BodySegmentation
- HandPose
- FaceMesh
- ImageClassifier
- SoundClassifier
- Sentiment

ml5 + Teachable Machine

Image + Teachable Machine

- Description
- Quick Start
- Examples
- Step-by-Step Guide
- Properties
- Methods

Train your own model!

NeuralNetwork

Learn

The ml5.js Image + Teachable Machine model allows you to create a model that can recognize the content of an image from a set of labels that you define. For example, you can train a model to tell the difference between a cat and a dog, happy and sad faces, or even between a hot dog and a sandwich.

The ml5.js Image + Teachable Machine model is a combination of the ml5.js imageClassifier and the

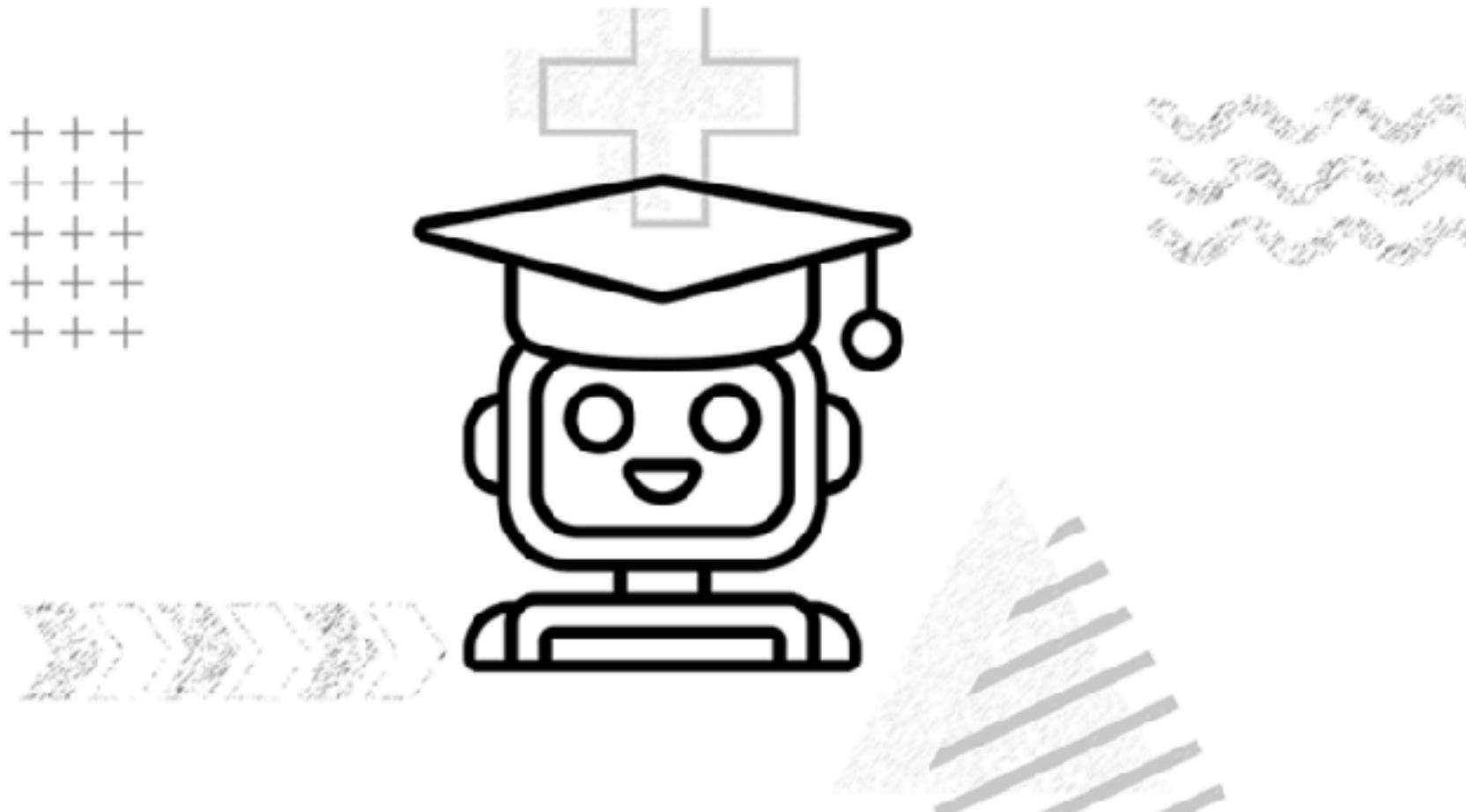


Image Credit: Juicy Fish | Contribute ❤️

roboflow

Product Solutions Resources Pricing Docs

Book a demo Sign in →

Everything you need to build and deploy computer vision models

Used by over 250,000 engineers to create datasets, train models, and deploy to production.

Get Started →

person 0.91

oven 0.51

chair 0.56

Upload Webcam Microsoft COCO ▾

github homepage-demo

OVER 250,000 DEVELOPERS AND MACHINE LEARNING ENGINEERS BUILD WITH ROBOFLOW

