

WebRTC 实时视频处理

——基础美颜与视频超分实践

高纯

声网 Agora
高级架构师

InfoQ官网 全新改版上线

促进软件开发领域知识与创新的传播



关注InfoQ网站
第一时间浏览原创IT新闻资讯



免费下载迷你书
阅读一线开发者的技术干货



自我介绍

高 纯

2019年至今 声网 Agora 高级架构师。

从事 Web Runtime 及 Web 平台音视频处理技术的研究工作。

2010-2018年 英特尔亚太研发中心开源技术中心

参与 MeeGo Browser、Chromium/Blink 内核、Crosswalk Web Runtime、Chrome OS、Android N 等多个开源 Web Engine 以及 OS 项目的研发。

目录

1. 应用背景与 HTML5 标准
2. Web 平台并行计算标准
3. 基于 WebGL 的基础美颜实现
4. Web 视频超分探索
5. Web 平台深度学习展望

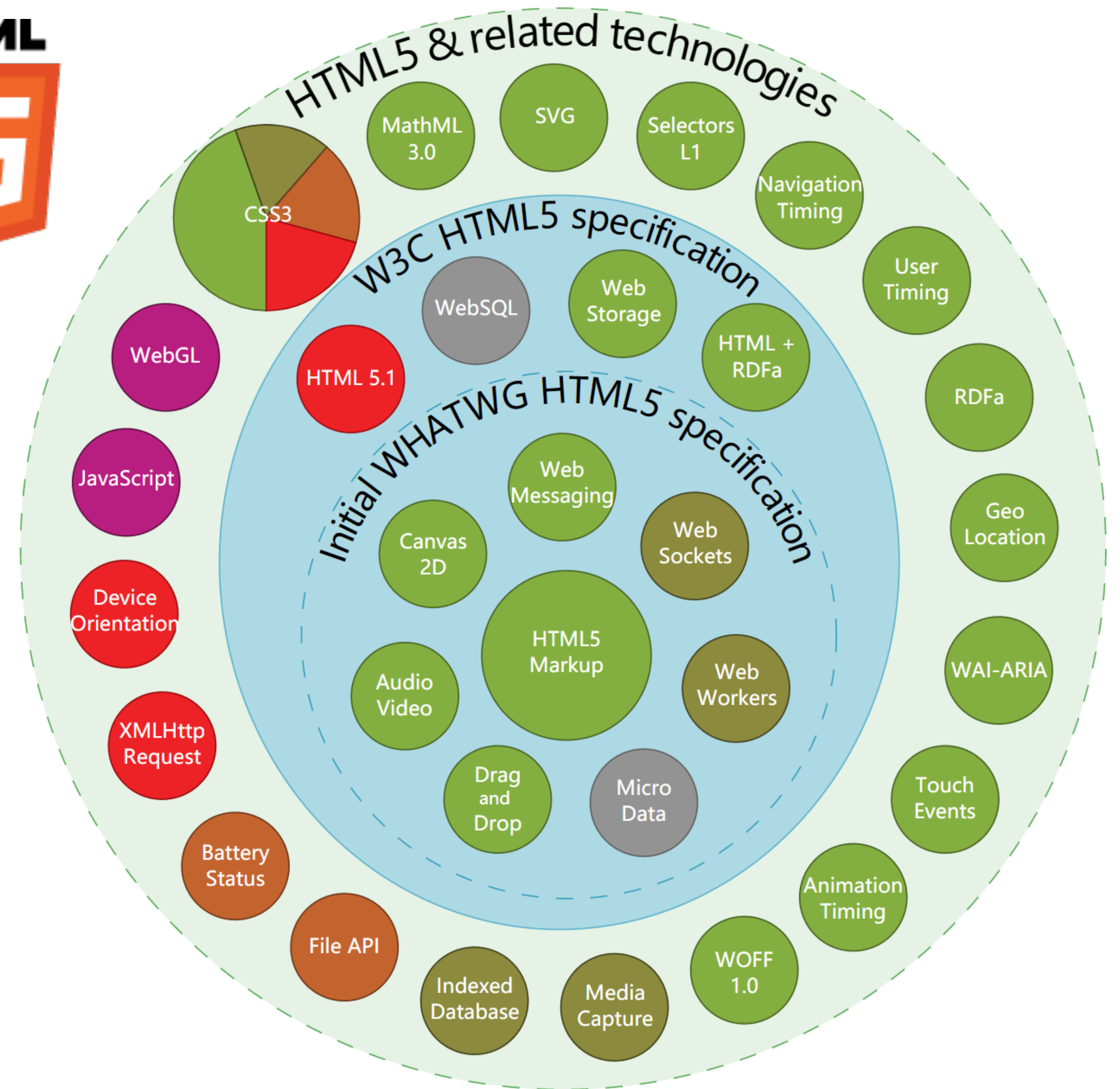
应用背景与HTML5标准

2014年10月，W3C发布HTML 5.0标准。随着主流浏览器逐步实现对HTML5新特性的支持，Web平台对多媒体内容的处理能力得到空前增强。WebRTC、WebGL、Web Audio等多媒体组件成为Web平台标配的能力。

近年来随着RTC、云游戏、人工智能等各种多媒体新需求的出现，HTML标准也在不断演进，WebVR、WebGPU、WebAssembly、WebGL Compute、WebNN等多项新的标准和规范正在被持续推动。

2019作为5G元年，RTC产业迎来蓬勃发展。如何通过HTML5技术为用户提过更多元、更友好的RTC服务，使Web平台成为既能快速触达又具客户粘滞性的渠道成为了RTC服务商所必须面对的问题。

以下将从WebRTC所承载的核心对象—实时视频流出发，通过2种典型的实时视频处理场景对Web平台面对的主要制约和潜在能力进行探索。



Web平台并行运算

OpenGL ES for the Web

基于OpenGL ES的跨平台Web 2D/3D图形标准，支持以JavaScript API使用GLSL。

WebGL 1.0基于OpenGL ES 2.0实现

WebGL 2.0基于OpenGL ES 3.0实现

WebGL 2.0 Compute基于OpenGL ES 3.1实现

WebGL支持的平台：



Chrome 9+
Chrome for **Android** 25+



Mozilla Firefox 4+
Firefox for **Android** YES



Safari 5.1+
Safari on **IOS** 8

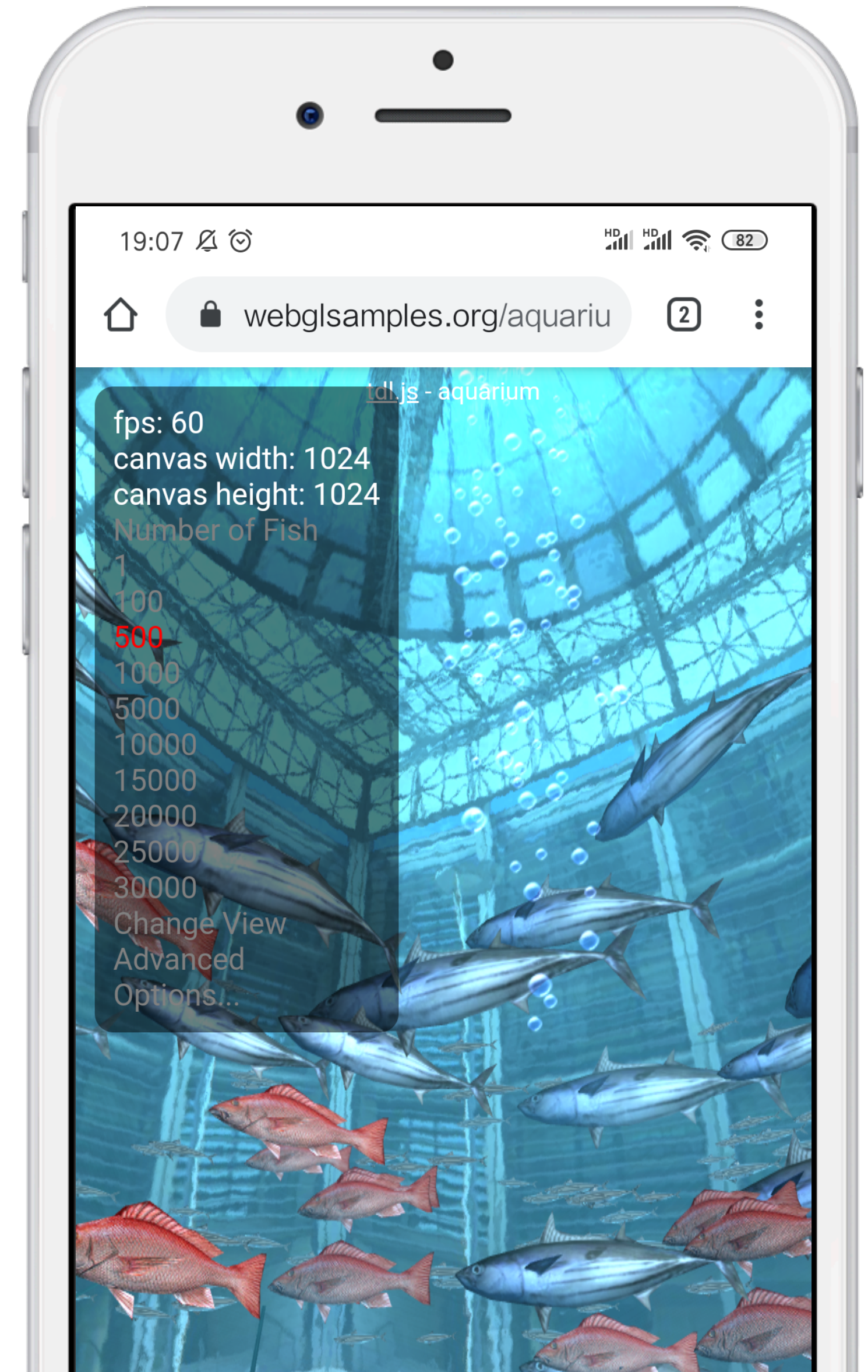


Microsoft Edge 12+

WebGL 2.0 Compute支持的平台：

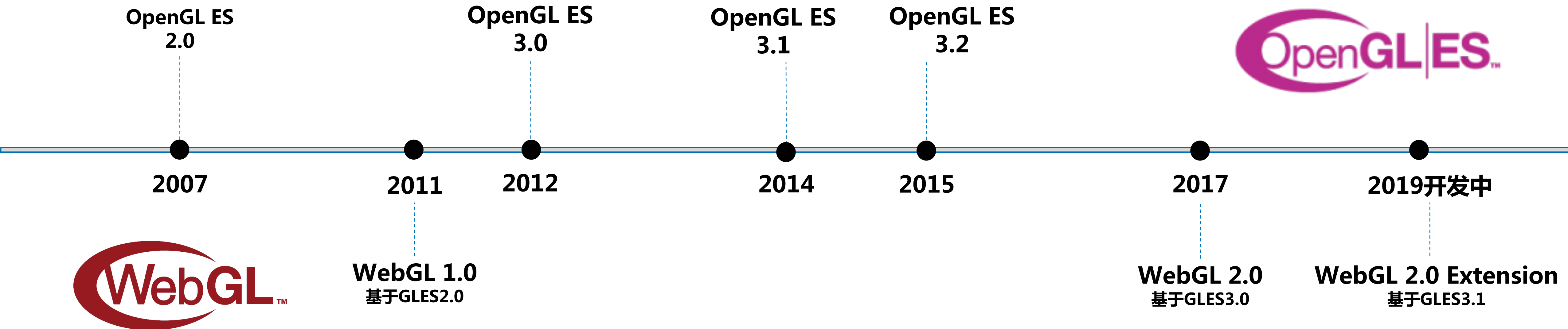


Chrome with experimental flag enabled



WebGL版本演进

WebGL与OpenGL ES的关系



WebGL 2.0 Extension包含：
WebGL 2.0 Compute
Parallel Shader Compilation
WEBGL_video_to_texture
.....

Web平台并行运算

WebGPU

目标

在Web平台提供新的接口来获取原生系统中现代3D图形及通用计算系统的能力。

底层实现：  

标准与实现

草案：<https://gpuweb.github.io/gpuweb/>

实现：[WebMetal](#) (Apple) [Dawn](#) (Google) [Obsidian](#) (Mozilla)

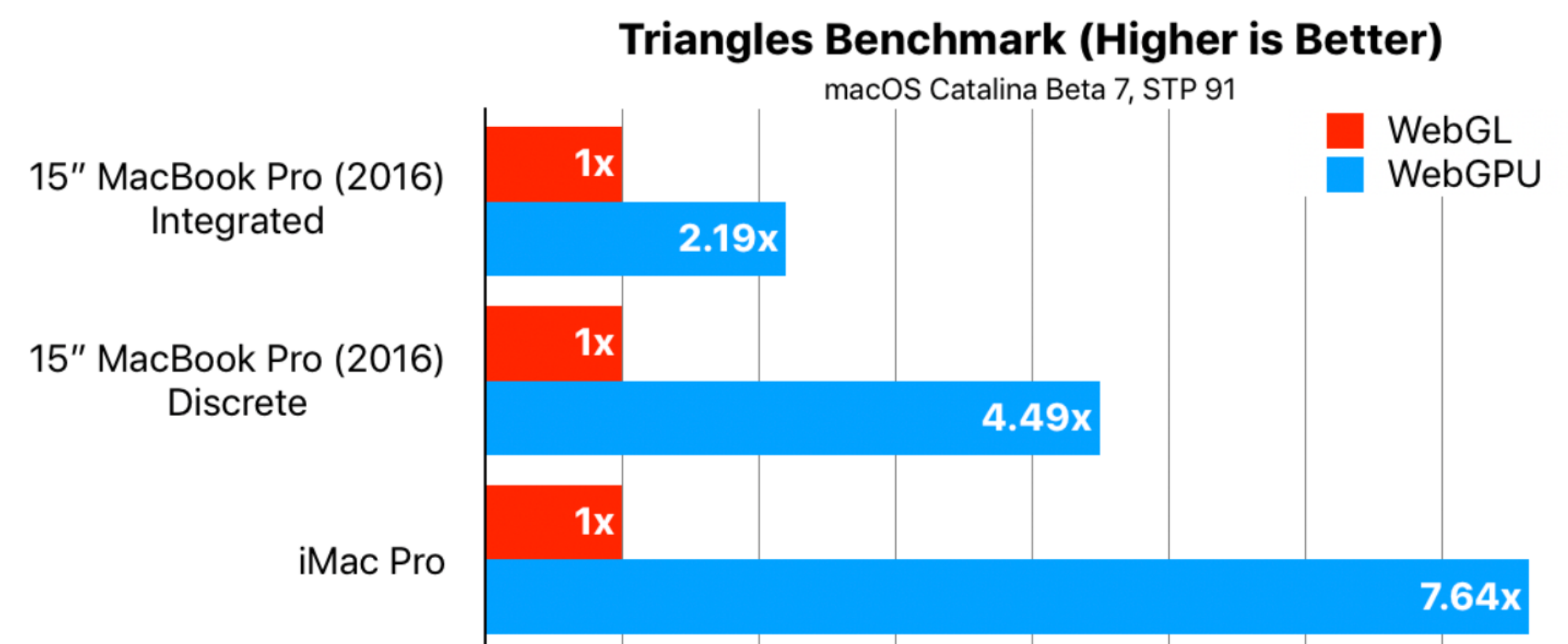
平台支持



Safari Tech Preview 91+



Chrome 78 for Mac
with experimental flag



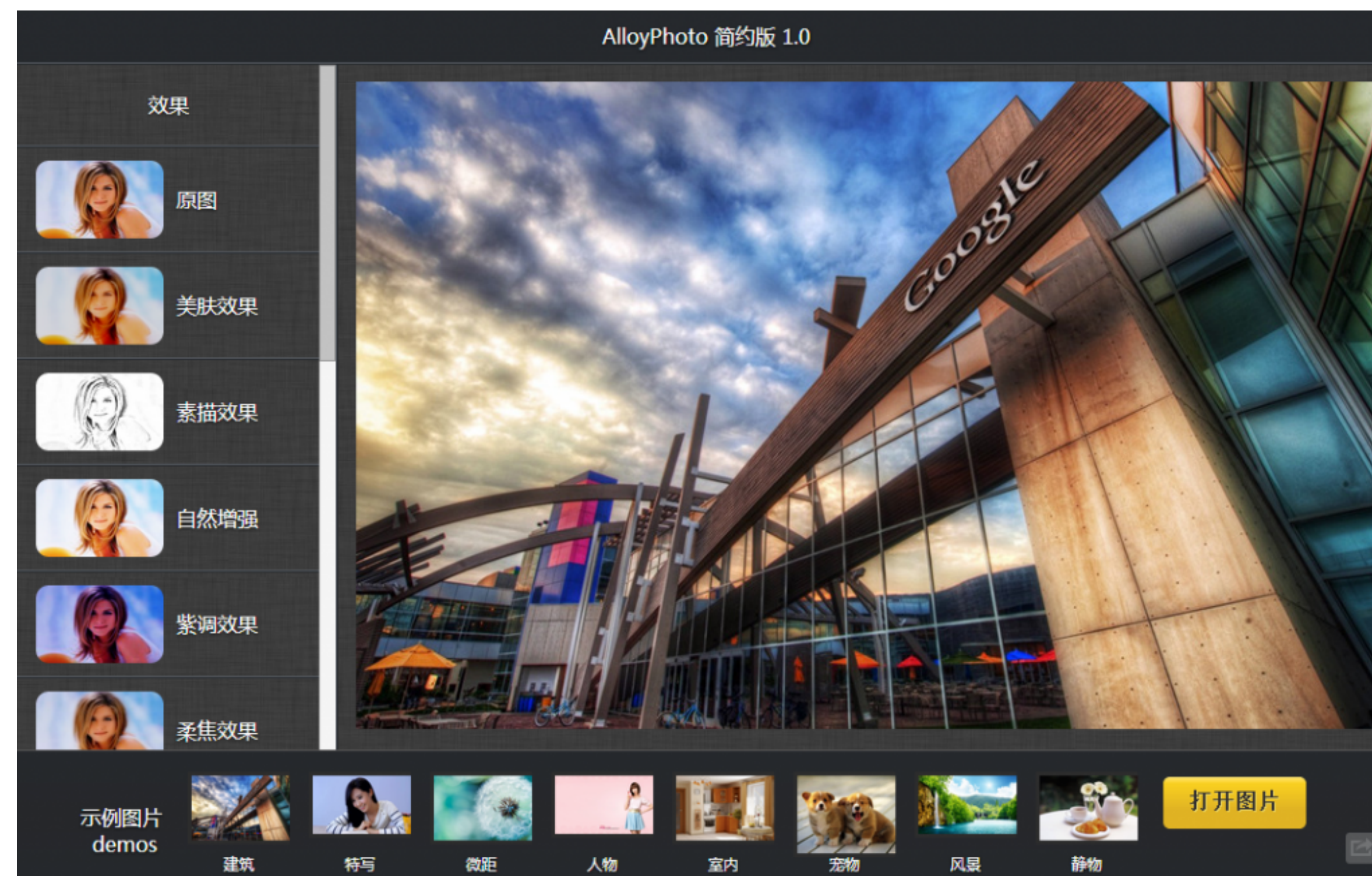
Web平台并行运算



Khronos OpenCL标准在Web平台的映射。依靠GPU和多核CPU提供多种并行处理能力，为图像和视频处理以及3D游戏物理运算提供加速。

在所有浏览器上均未实现；

Intel、Samsung、Nokia分别实现过通过浏览器插件或Web Runtime实现过自己的WebCL版本。



Tencent AlloyImage

Web平台并行运算

	性能	平台兼容性	文档&易用性
	较好	最好👍	较好
	N/A	最差	较差
WebGPU	最好👍	较差	较差

基于WebGL的基础美颜实现

基础美颜使用的相关Web模块

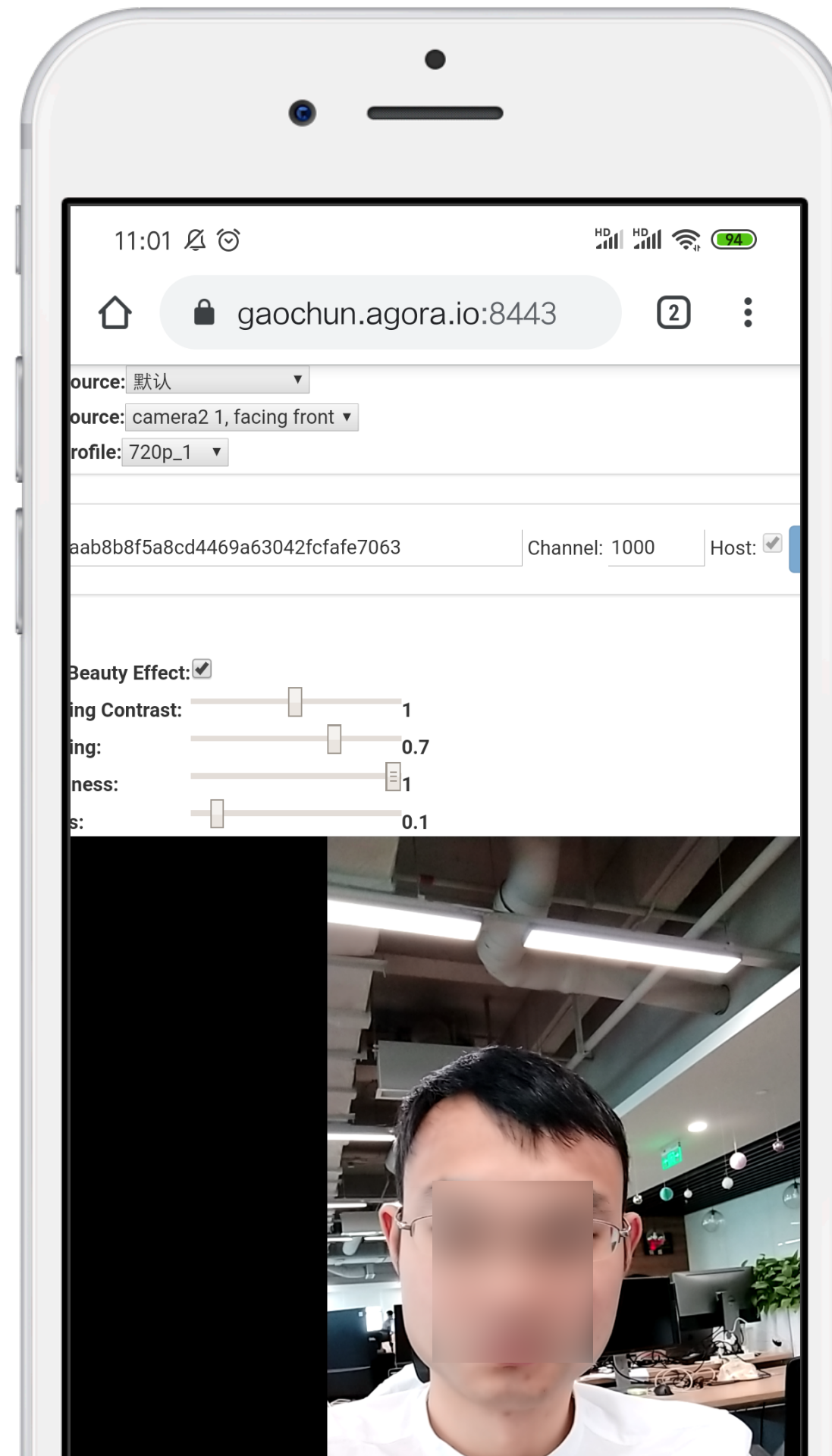
视频采集、处理及传输



Camera数据采集及网络媒体传输



GLSL实现图像滤波算法



Canvas

视频及图像渲染



Agora Web SDK

传输控制、RTC框架

基于WebGL的基础美颜实现

主要步骤

1. 视频采集

```
var constraints = { audio: true, video: { width: 1280, height: 720 } };
var video;

navigator.mediaDevices.getUserMedia(constraints)
  .then(function(mediaStream) {
    video = document.querySelector('video');
    video.srcObject = mediaStream;
    video.onloadedmetadata = function(e) {
      video.play();
    };
  })
  .catch(function(err) { console.log(err.name + ": " + err.message); });
```

2. 创建WebGL纹理并应用GLSL

```
var canvas = document.getElementById('myCanvas');
var gl = canvas.getContext('webgl2');
.....
gl.texSubImage2D(gl.TEXTURE_2D, 0, 0, gl.RGBA, gl.RGBA,
                 gl.UNSIGNED_BYTE, video);
.....
gl.useProgram(skinToneProgram);
.....
gl.drawArray(gl.TRIANGLES, 0, 6);
```

3. 从Canvas获取MediaStream

```
var fps = localStream.attributes.maxFrameRate;

var canvasCaptureMediaStream = canvas.captureStream(fps);
var videoTrack = canvasCaptureMediaStream.getVideoTracks()[0];
```

4. 发送VideoTrack

```
var rtcRtpSender = that.pc.peerConnection.getSenders();

rtcRtpSender.replaceTrack(videoTrack);
```

基于WebGL的基础美颜实现

WebGL美颜处理帧率

设备		FPS
Redmi Note7 骁龙660	480p	58
Redmi Note7 骁龙660	720p	29
Redmi Note7 骁龙660	1080p	7
PC Intel i7 UHD620集显	480p	60
PC Intel i7 UHD620集显	720p	60
PC Intel i7 UHD620集显	1080p	22

 [视频演示效果](#)

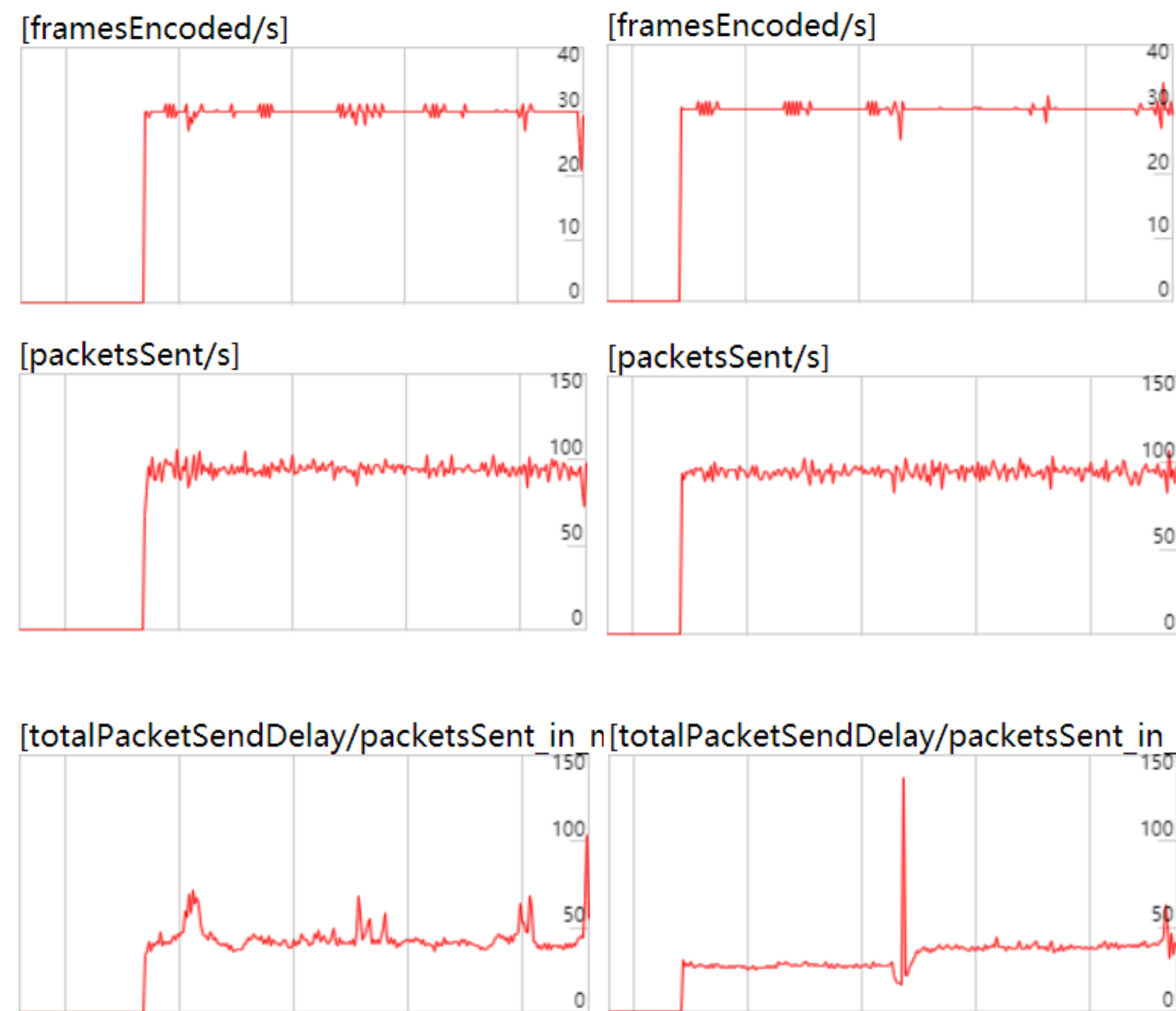
基于WebGL的基础美颜实现

实时美颜处理对WebRTC性能影响

480p 30
FPS

关闭美颜

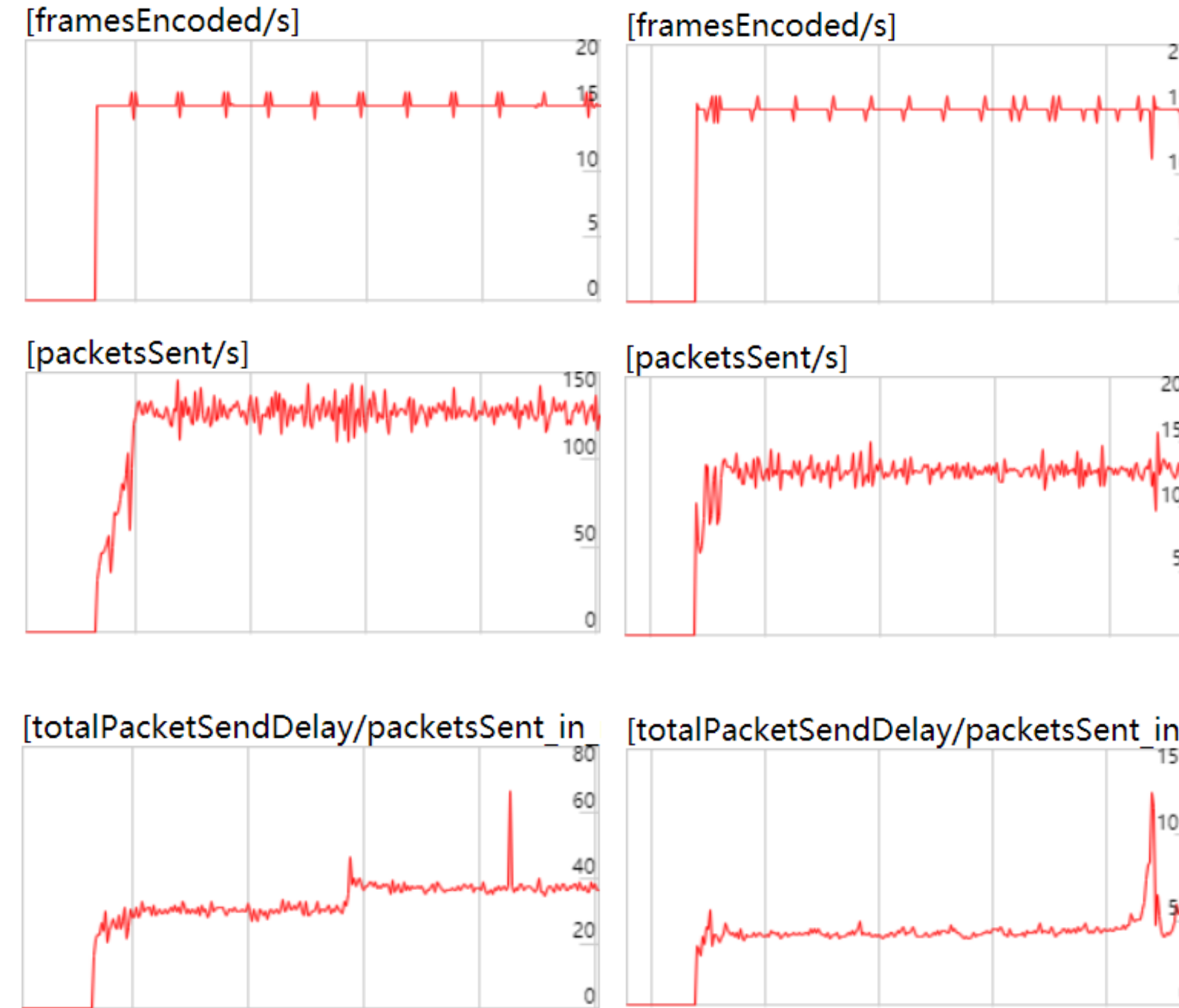
开启美颜



720p 15
FPS

关闭美颜

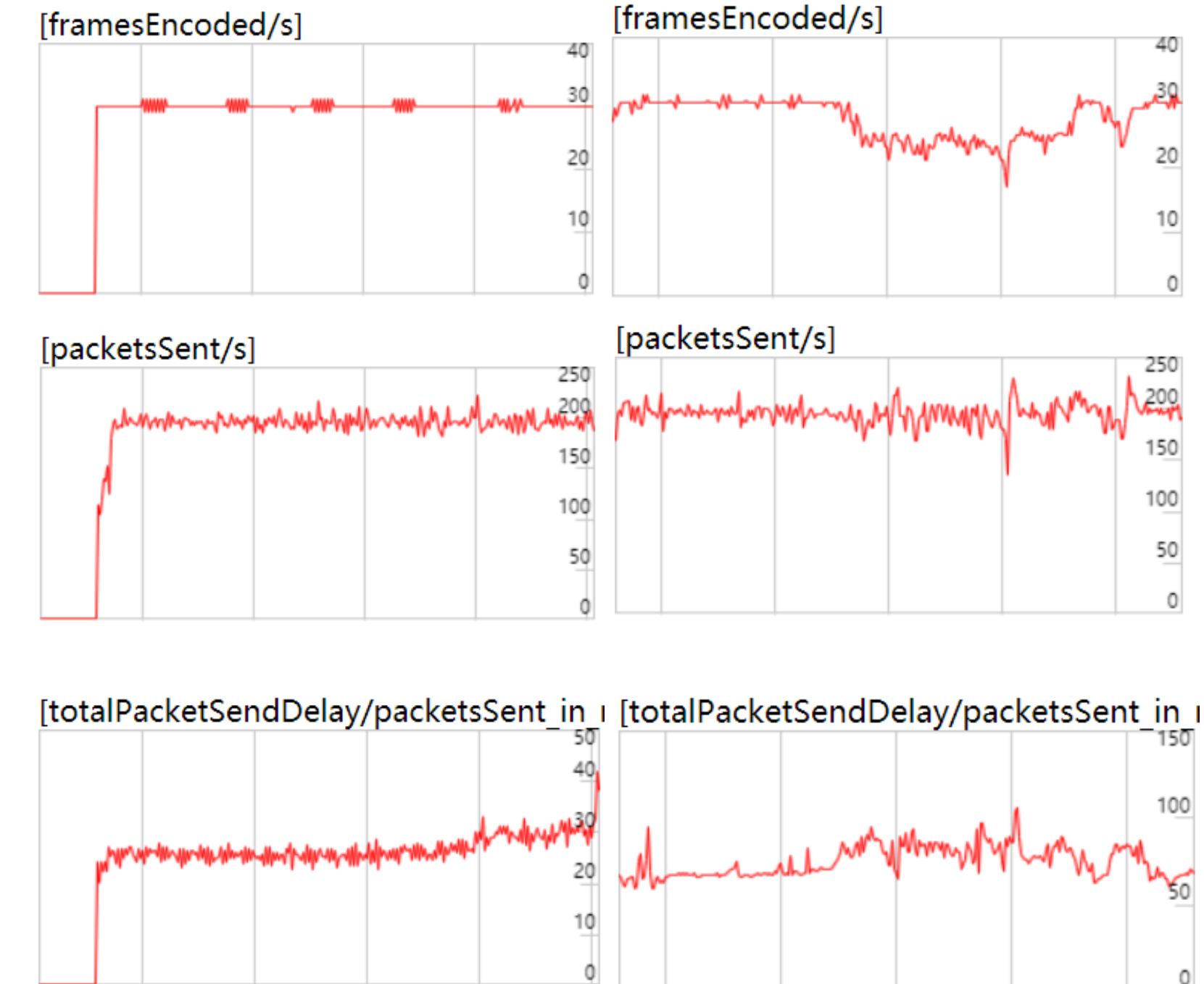
开启美颜



720p 30
FPS

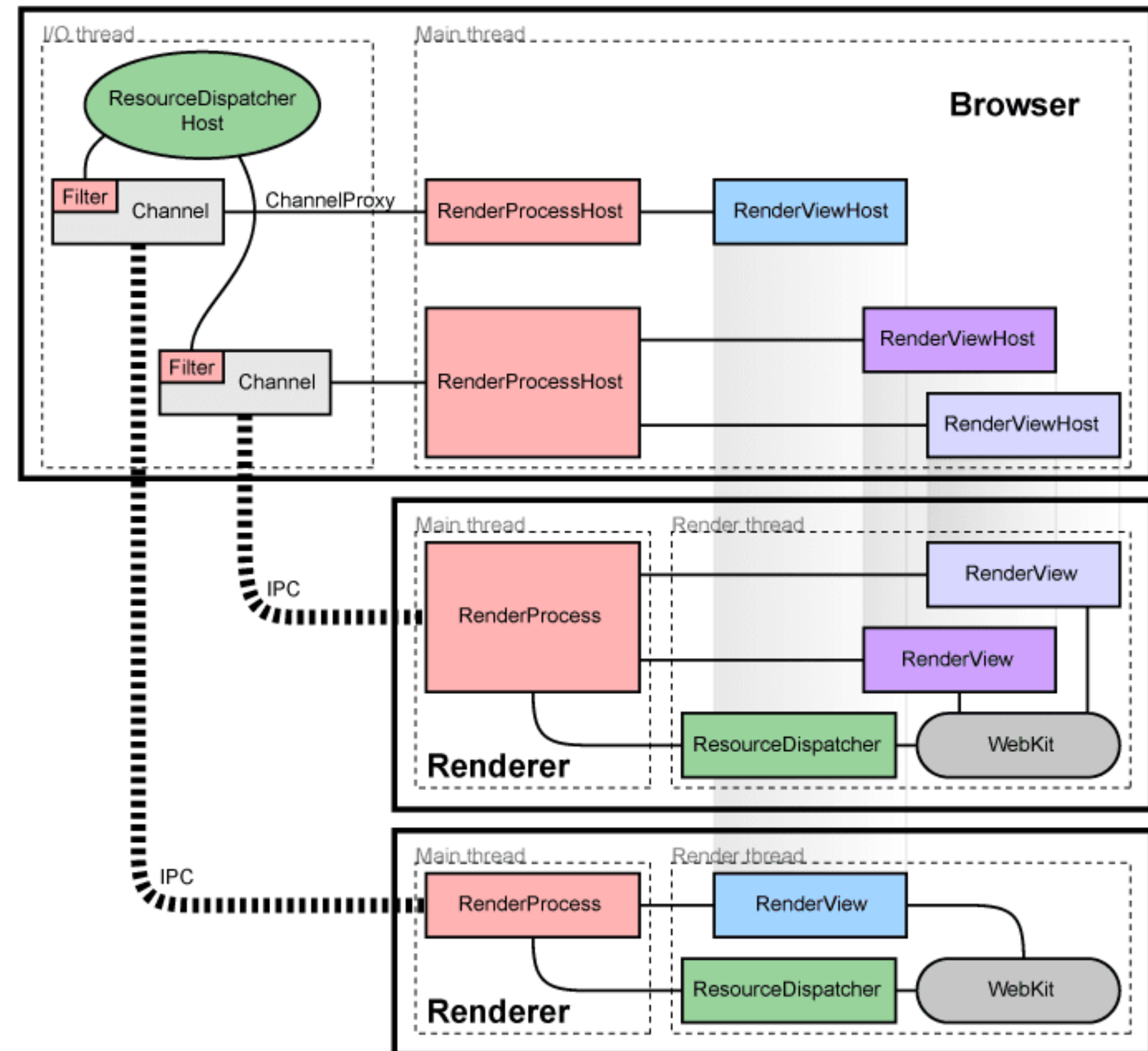
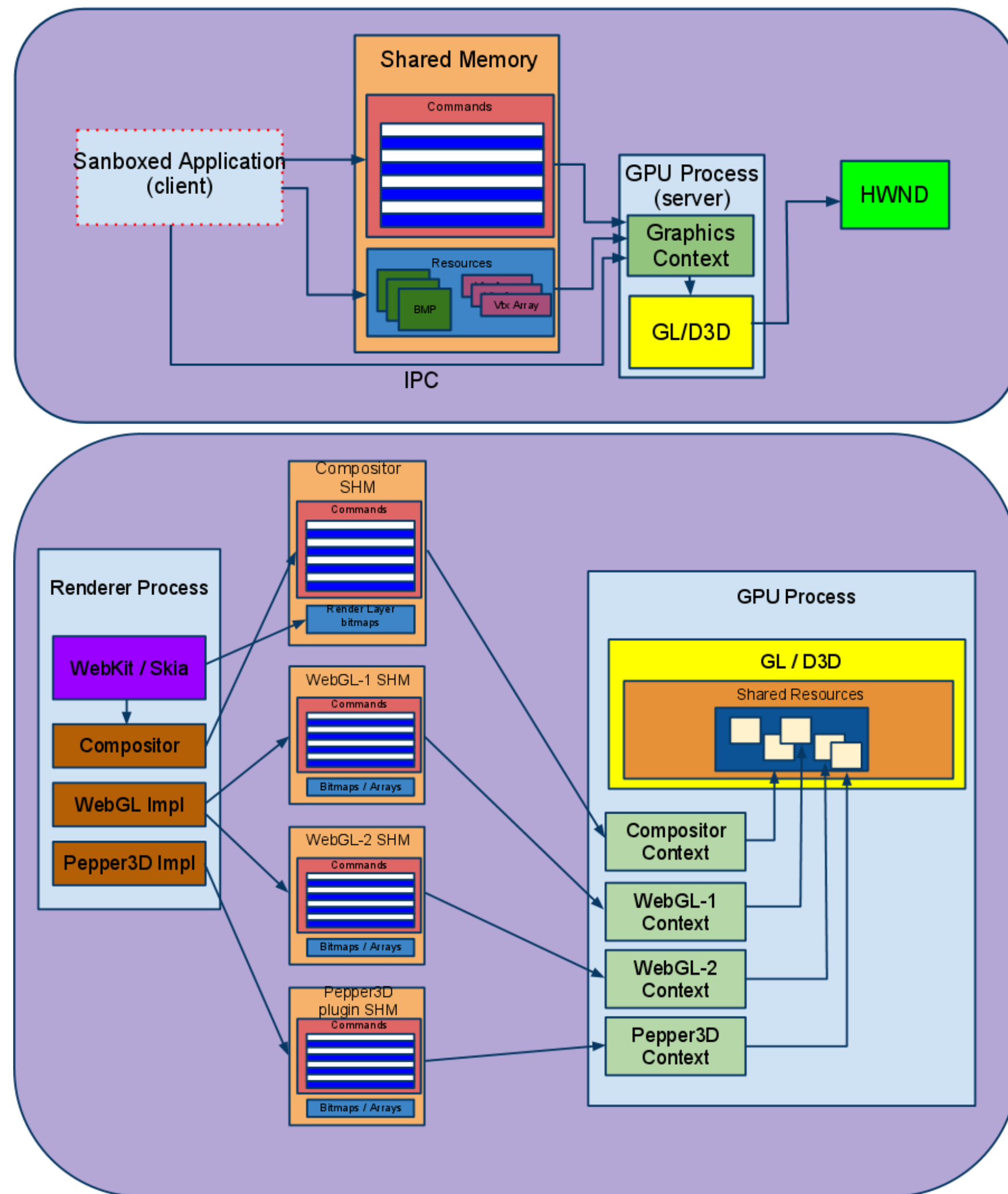
关闭美颜

开启美颜



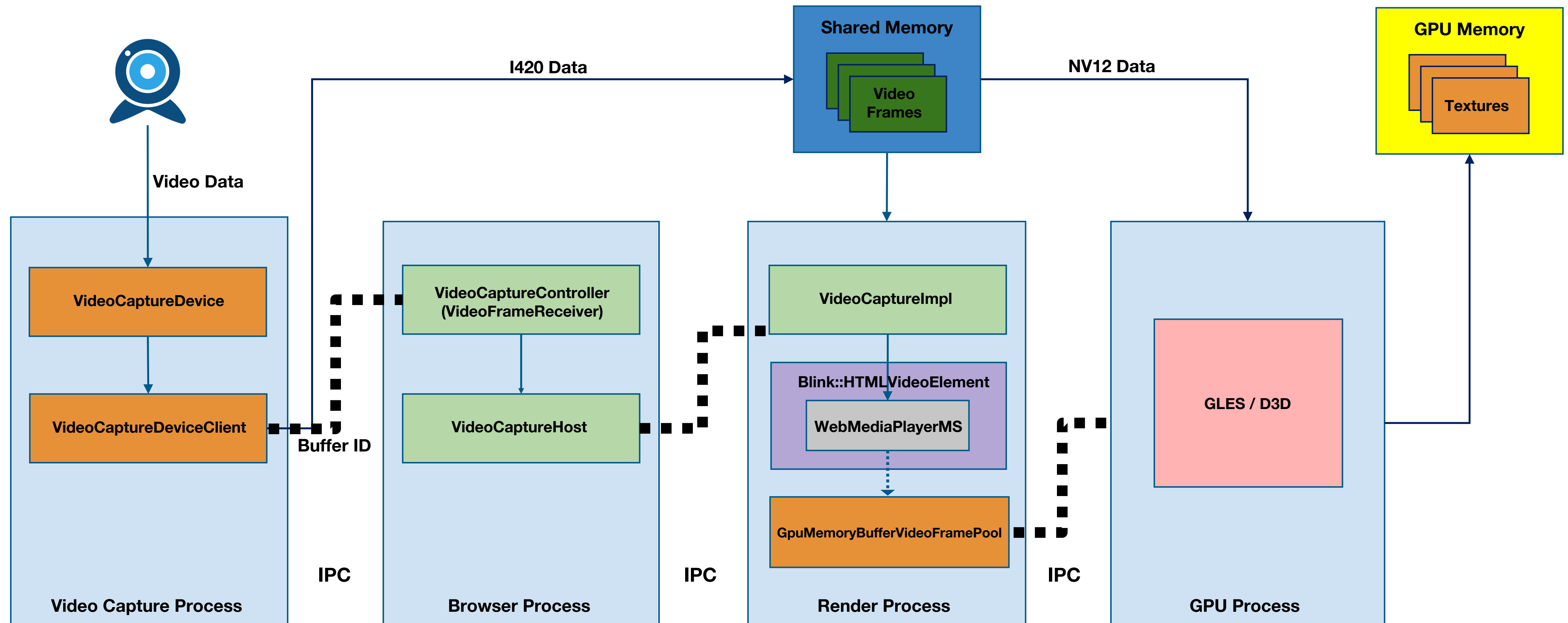
基于WebGL的基础美颜实现

Chromium多进程与硬件加速合成机制



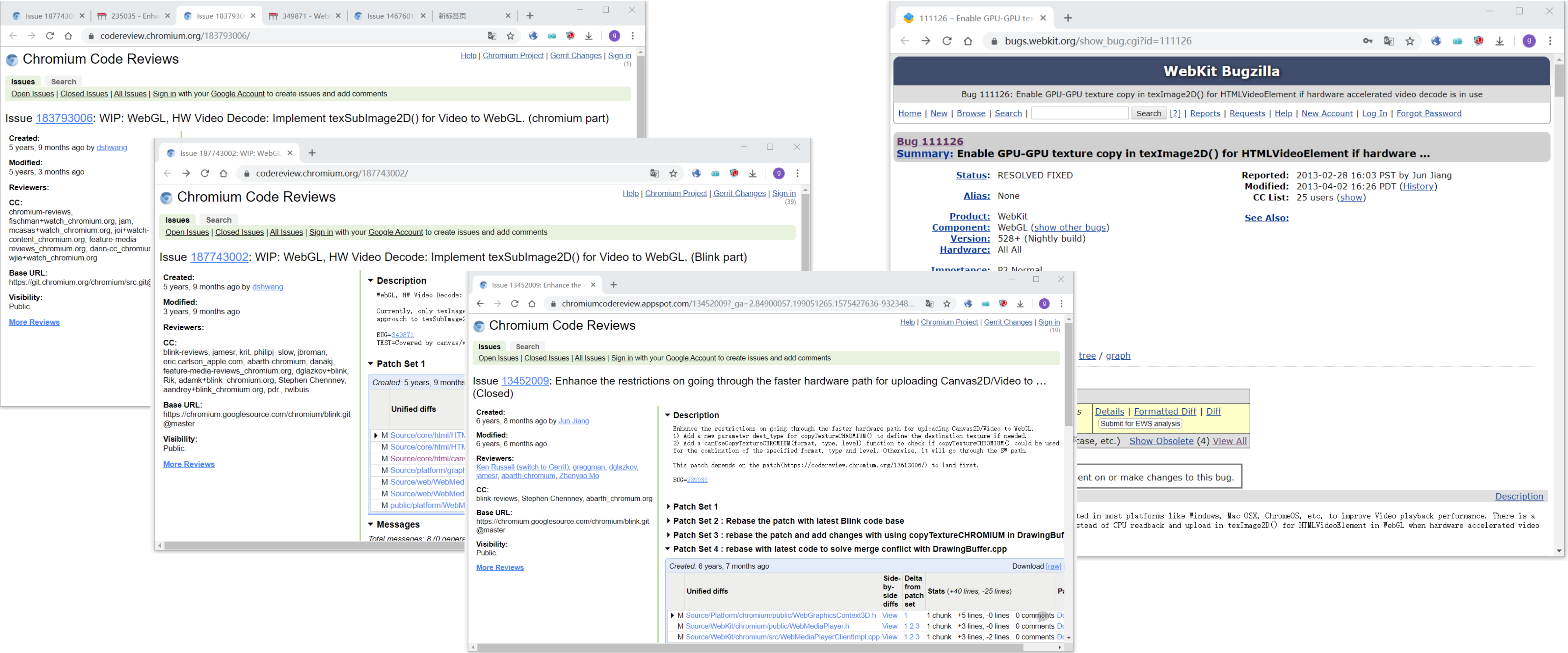
基于WebGL的基础美颜实现

Chromium Camera视频采集与渲染



基于WebGL的基础美颜实现

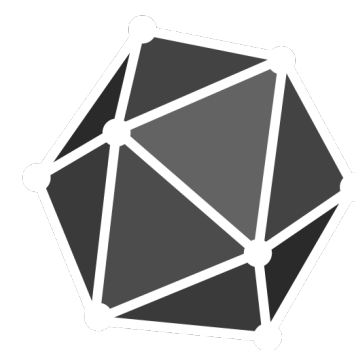
Chromium从Video生成WebGL纹理的性能优化



WebGL视频超分探索

SRCNN工具箱

Web平台开源CNN框架



ONNX.js

支持onnx格式的深度学习模型文件
WebGL加速
实现了基本的Operation



Keras.js

支持Keras深度学习模型文件
WebGL加速
项目在Tensorflow.js发布后停止维护



TensorFlow.js

支持Tensorflow和Keras格式的模型文件
WebGL加速
支持较多的Operation

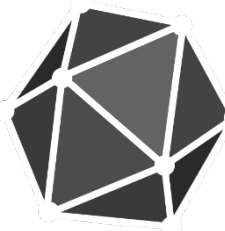


MIL WebDNN

实现了web平台对Keras深度学习模型的支持
WebGPU或WebAssembly实现
WebGPU计算只能在MacOS或IOS上运行

WebGL视频超分探索

SRCNN工具箱对比

	GPU加速	平台兼容性	模型兼容性	生态及社区
 ONNX.js	WebGL	好	一般	一般
 Keras.js	WebGL	好	较好	停止维护
 TensorFlow.js	WebGL	好	较好	活跃
 MIL WebDNN	WebGPU	较差	一般	一般

WebGL视频超分探索

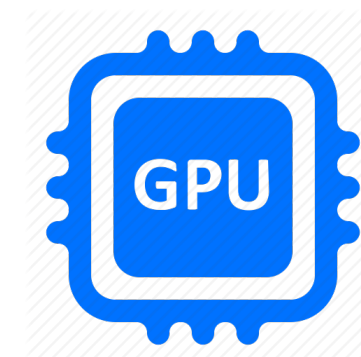
TensorFlow.js



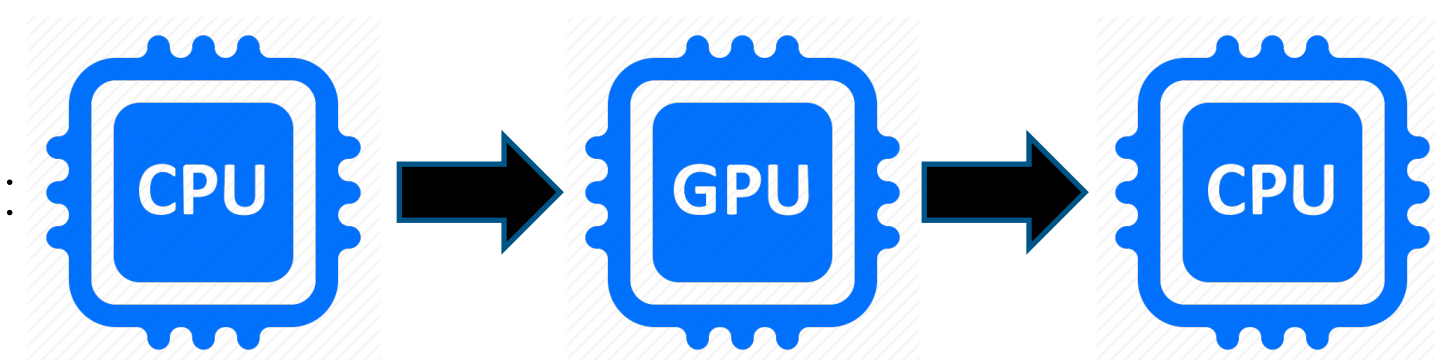
在浏览器和Node.js训练和部署机器学习模型的通用JavaScript库。

- TensorFlow生态成员，获TensorFlow的工具生态系统支持
- 具有WebGL和WebAssembly两种后端，计划实现WebGPU后端
- 支持Keras模型及TensorFlow GraphDef模型导入，可通过第三方工具导入Onnx模型文件
- 无法支持大规模、长时间训练，以及多机分布式训练

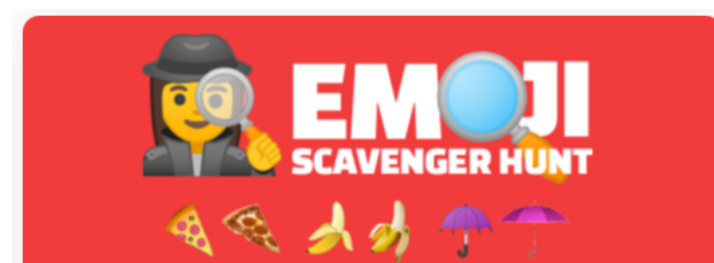
WebGL推理:



数据IO过程:



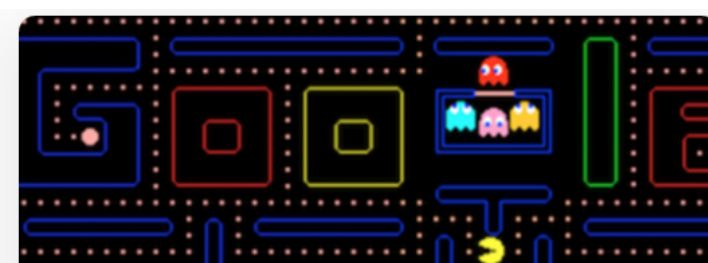
运算特性



Emoji Scavenger Hunt

使用手机摄像头识别现实世界中的表情符号。您能在时间结束前找到所有的表情符号吗？

[查看代码](#)



网络摄像头控制器

在浏览器中使用训练过的图像玩《吃豆人》游戏。

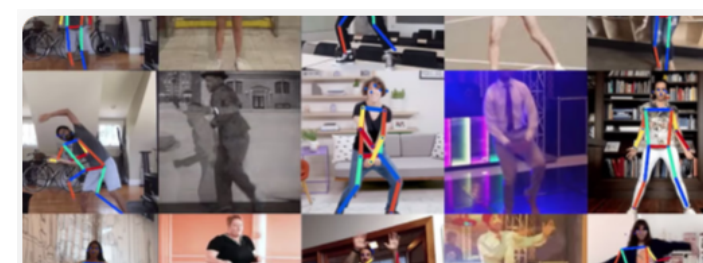
[查看代码](#)



Teachable Machine

无需编码！教机器识别图像和播放声音。

[查看代码](#)



Move Mirror

只需四处走动，就能以有趣的新方式探索图片。

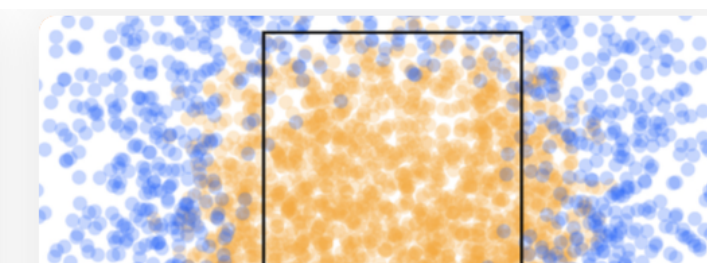
[查看代码](#)



性能 RNN

欣赏神经网络的现场钢琴演奏。

[查看代码](#)



Node.js 音高预测

使用 Node.js 训练一个服务器端模型以对棒球投掷类型进行分类。

[查看代码](#)

WebGL视频超分探索

基于TensorFlow.js实现超分

视频数据获取

```
var canvas = document.getElementById('canvas');  
var gl = canvas.getContext('2d');  
gl.drawImage(video, 0, 0);  
  
var imageData = gl.getImageData(0, 0, width, height);
```

神经网络模型加载和推理

```
var model = await tf.loadLayersModel('sr-tf/model.json');  
const inputTensor = tf.tensor(inputs, [1, 1, height, width], 'float32');  
model.summary();  
const prediction = model.predict(inputTensor);  
const outputY = await prediction.data();
```

超分图像显示

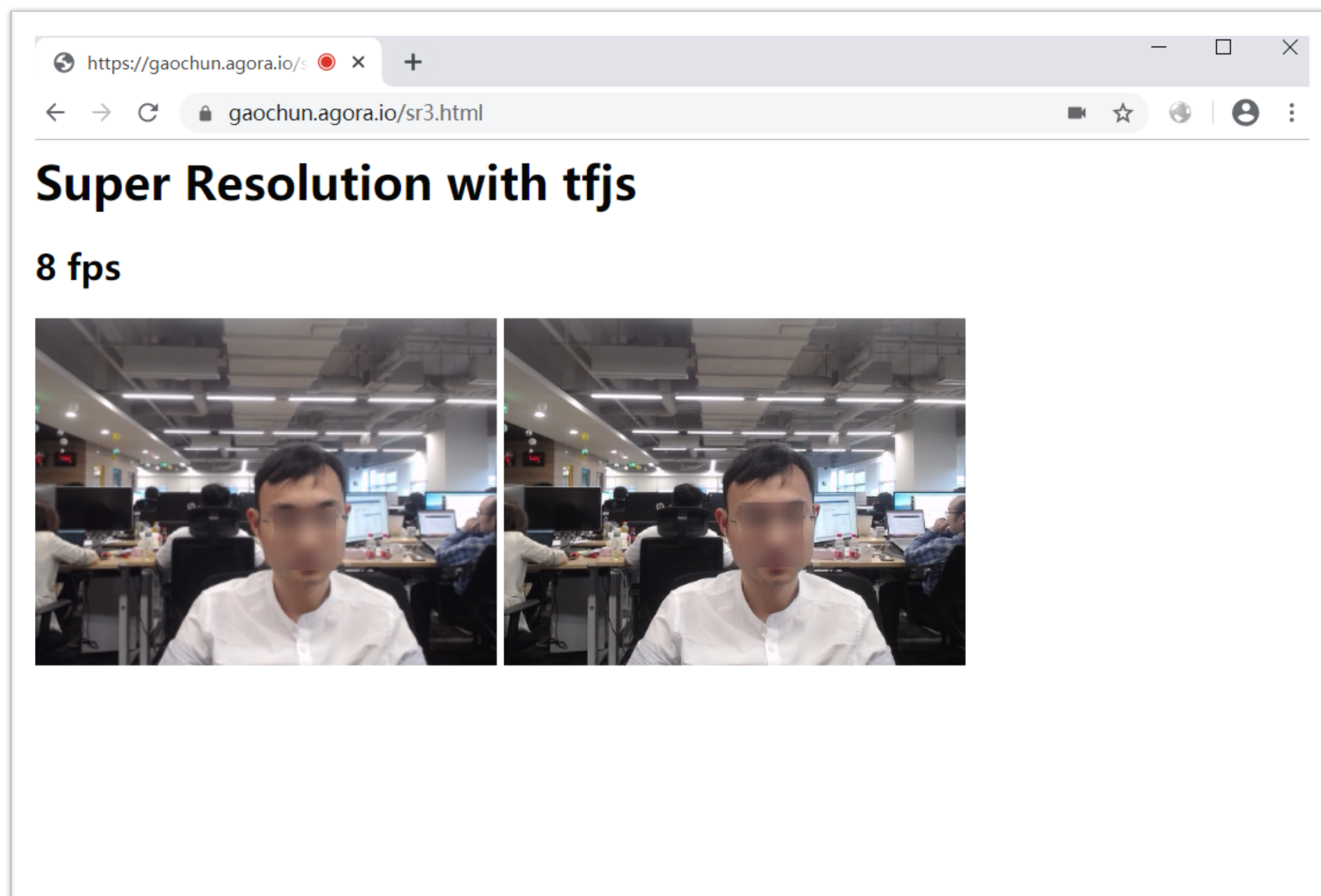
```
var srlImage = new ImageData(srDataArray, width * 2, height * 2);  
document.getElementById('tf-canvas').getContext('2d').putImageData(srlImage, 0, 0);
```

WebGL视频超分探索

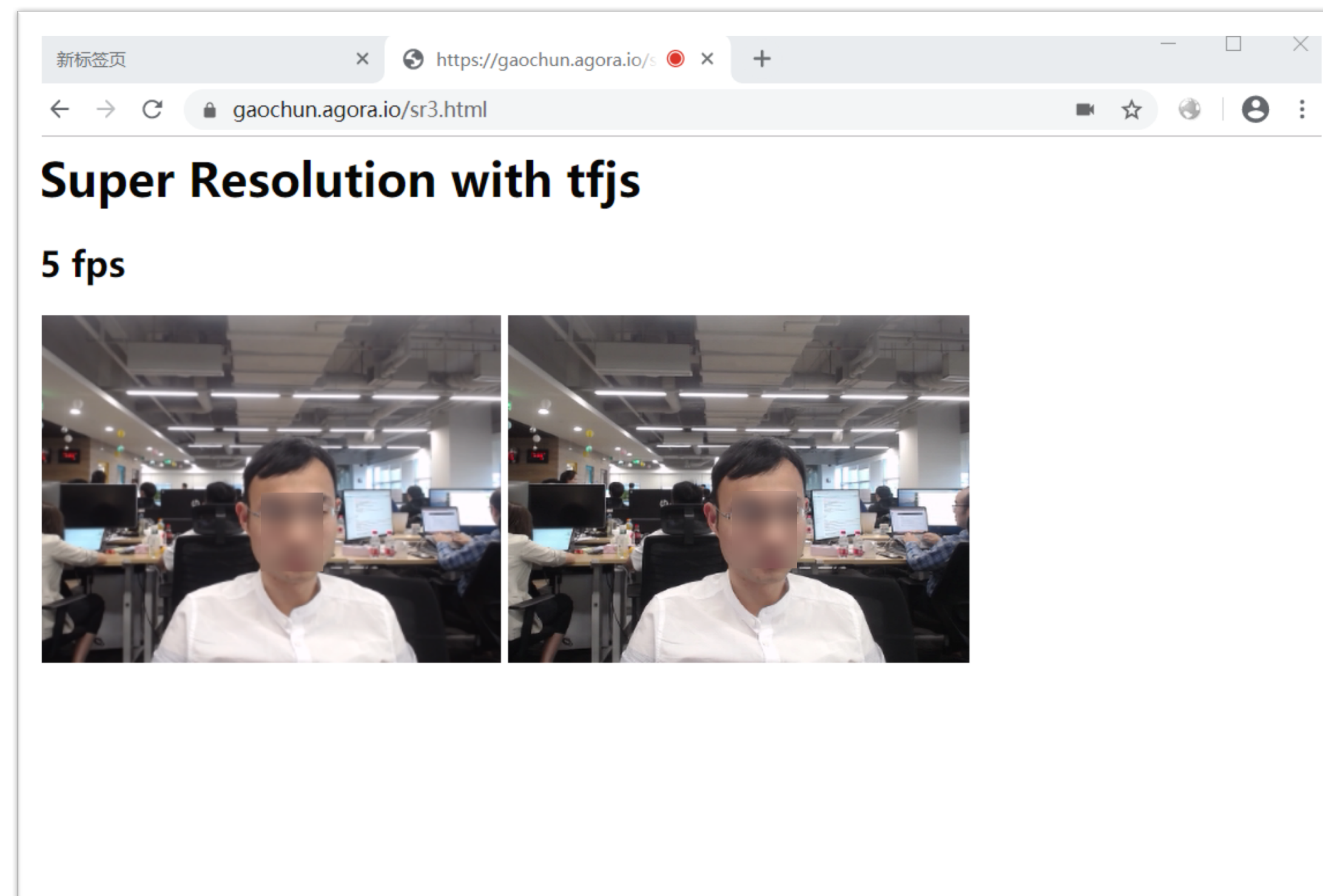
TensorFlow.js视频超分帧率

视频分辨率：240 x 320

处理方式：逐帧处理



 **Geforce MX250**
NVIDIA®



 **UHD Graphics 620**

WebGL视频超分探索

TensorFlow.js接口定制

增加从Video生成纹理的输入接口

该接口以HTMLVideoElement为参数，直接将从Video元素生成texture，避免从Javascript输入图像数据。

增加推理结果的纹理获取接口

通过该接口应用可以直接从tensorflow.js拿到Texture推理结果，应用可以通过WebGL对结果继续进行处理直至在Canvas中显示，避免超分数据在GPU和CPU之间的复制。

```
fromElement(  
  pixels: PixelData|ImageData|HTMLImageElement|HTMLCanvasElement|  
  HTMLVideoElement,  
  numChannels: number,  
  callback: (gpgpu: any) => void): Tensor3D {  
  
  .....  
  const tempPixelHandle = this.makeTensorHandle(texShape, 'int32');  
  this.texData.get(tempPixelHandle.dataId).usage = TextureUsage.PIXELS;  
  this.gpgpu.uploadPixelDataToTexture(  
    this.getTexture(tempPixelHandle.dataId), pixels as ImageData);  
  this.gpgpu.inputTexture = this.getTexture(tempPixelHandle.dataId);  
  let program, res;  
  if (ENV.getBool('WEBGL_PACK')) {  
    program = new FromPixelsPackedProgram(outShape);  
    const packedOutput =  
      this.makeOutputArray(program.outputShape, 'float32');  
    res = this.compileAndRun(program, [tempPixelHandle], packedOutput);  
  } else {  
    program = new FromPixelsProgram(outShape);  
    res = this.compileAndRun(program, [tempPixelHandle]);  
  }  
  this.disposeData(tempPixelHandle.dataId);  
  callback(this.gpgpu);  
  return res as Tensor3D;  
}
```

tfjs-core/src/backends/webgl/backend_webgl.ts

WebGL视频超分探索

基于定制tfjs的视频超分实现

```
function sr(video) {  
  // Create input tensor from Video element  
  var videoInput = tf.browser.fromElement(video, 1,  
                                           function(result) {  
      gpgpu = result;  
    });  
  videoInput.shape = [1, 1, 240, 320];  
  videoInput.strides = [76800, 76800, 320];  
  
  // Predict and create output texture  
  const prediction = model.predict(videoInput);  
  
  // Make output texture drawable and draw to canvas  
  var aLocation = gl.getUniformLocation(program, "A");  
  var bLocation = gl.getUniformLocation(program, "B");  
  
  gl.bindFramebuffer(gl.FRAMEBUFFER, null);  
  gl.viewport(0, 0, width * 2, height * 2);
```

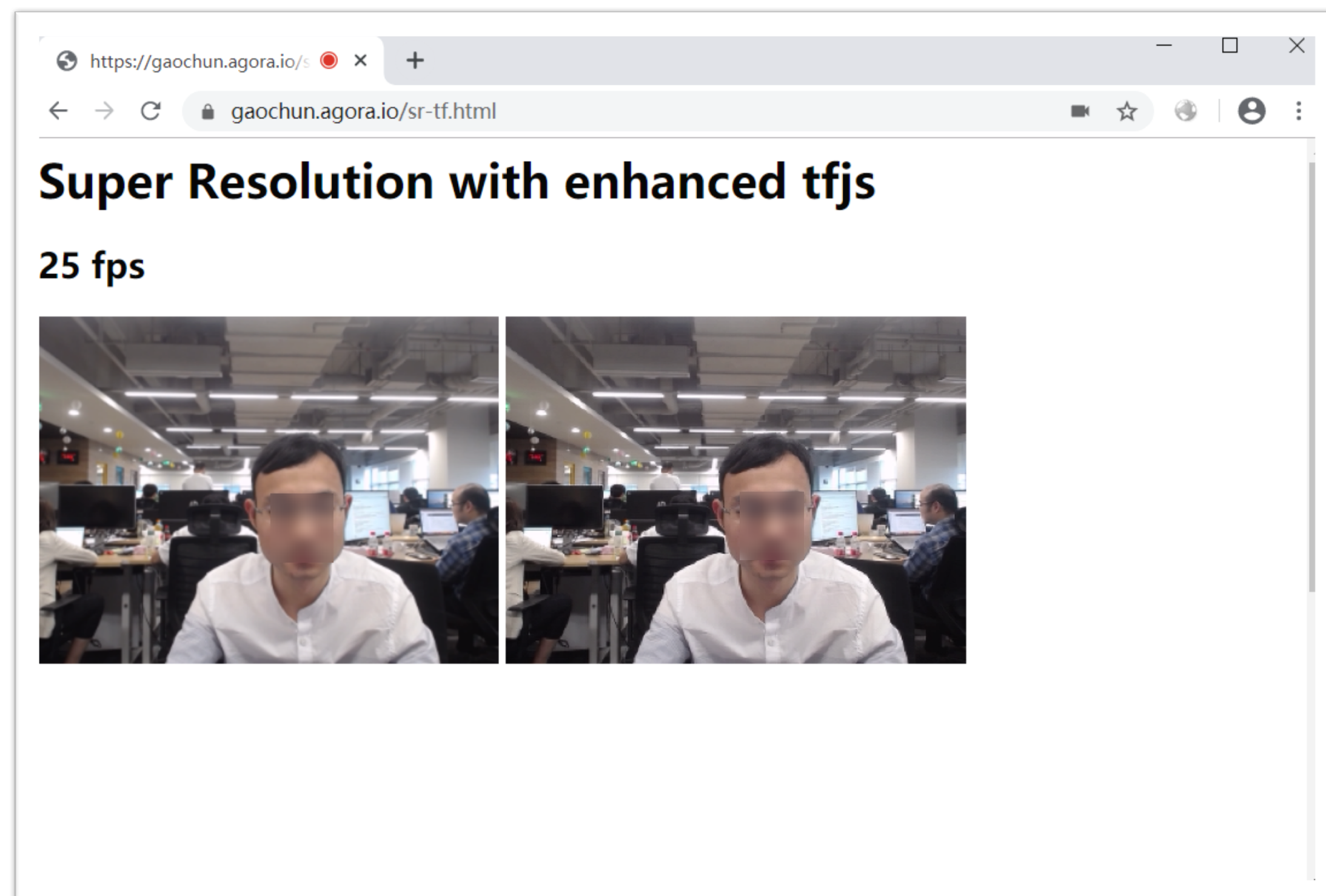
```
  gl.scissor(0, 0, width * 2, height * 2);  
  
  gl.useProgram(screenProgram.program);  
  
  gl.activeTexture(gl.TEXTURE0);  
  gl.bindTexture(gl.TEXTURE_2D, gpgpu.outputTexture);  
  gl.uniform1i(aLocation, 0);  
  
  gl.activeTexture(gl.TEXTURE1);  
  gl.bindTexture(gl.TEXTURE_2D, gpgpu.inputTexture);  
  gl.uniform1i(bLocation, 1);  
  
  gl.drawElements(gl.TRIANGLES, 6, gl.UNSIGNED_SHORT, 0);  
  
  // Release tensors  
  videoInput.dispose();  
  prediction.dispose();  
}
```


WebGL视频超分探索

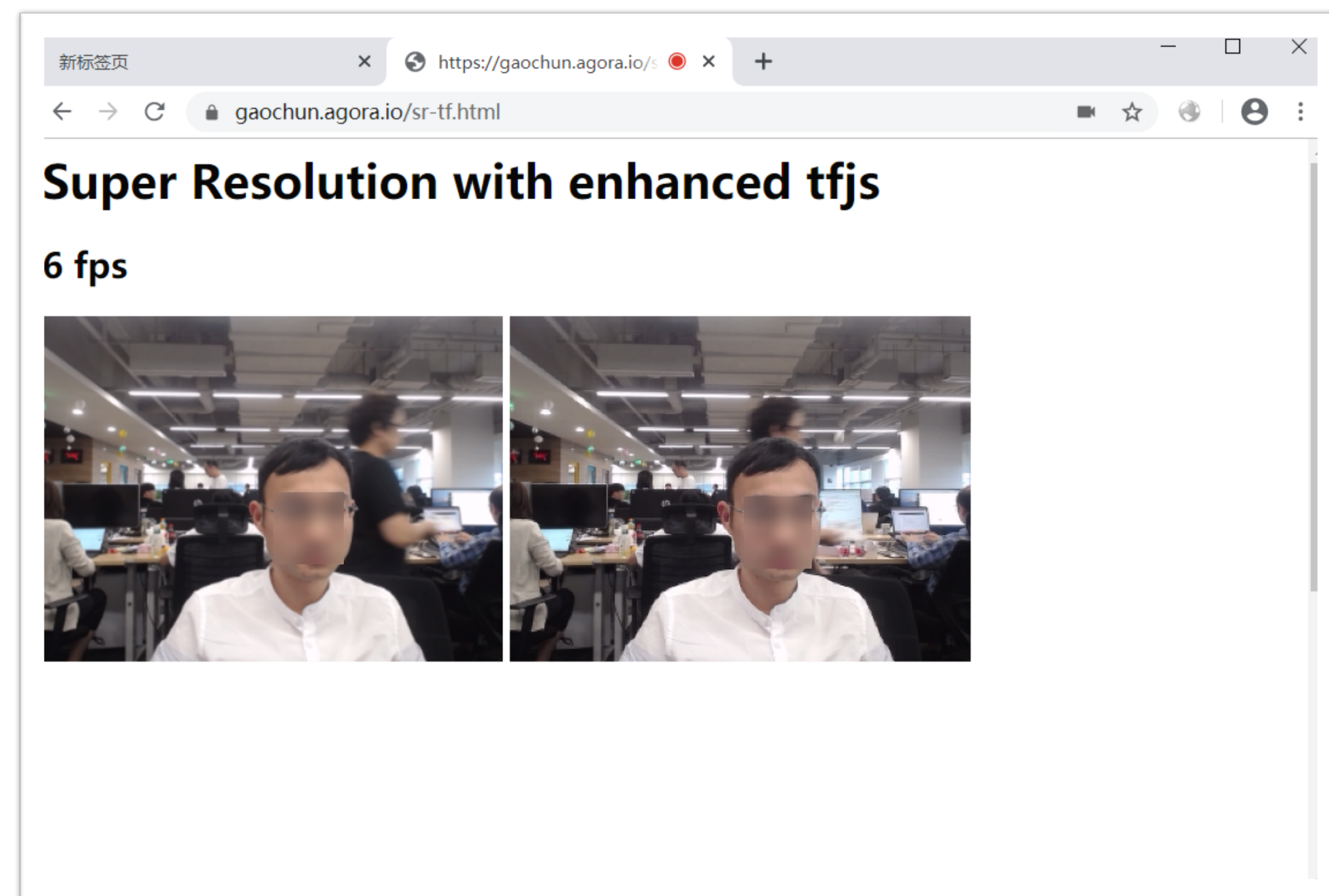
基于定制tfjs的视频超分帧率

视频分辨率：240 x 320

处理方式：逐帧处理



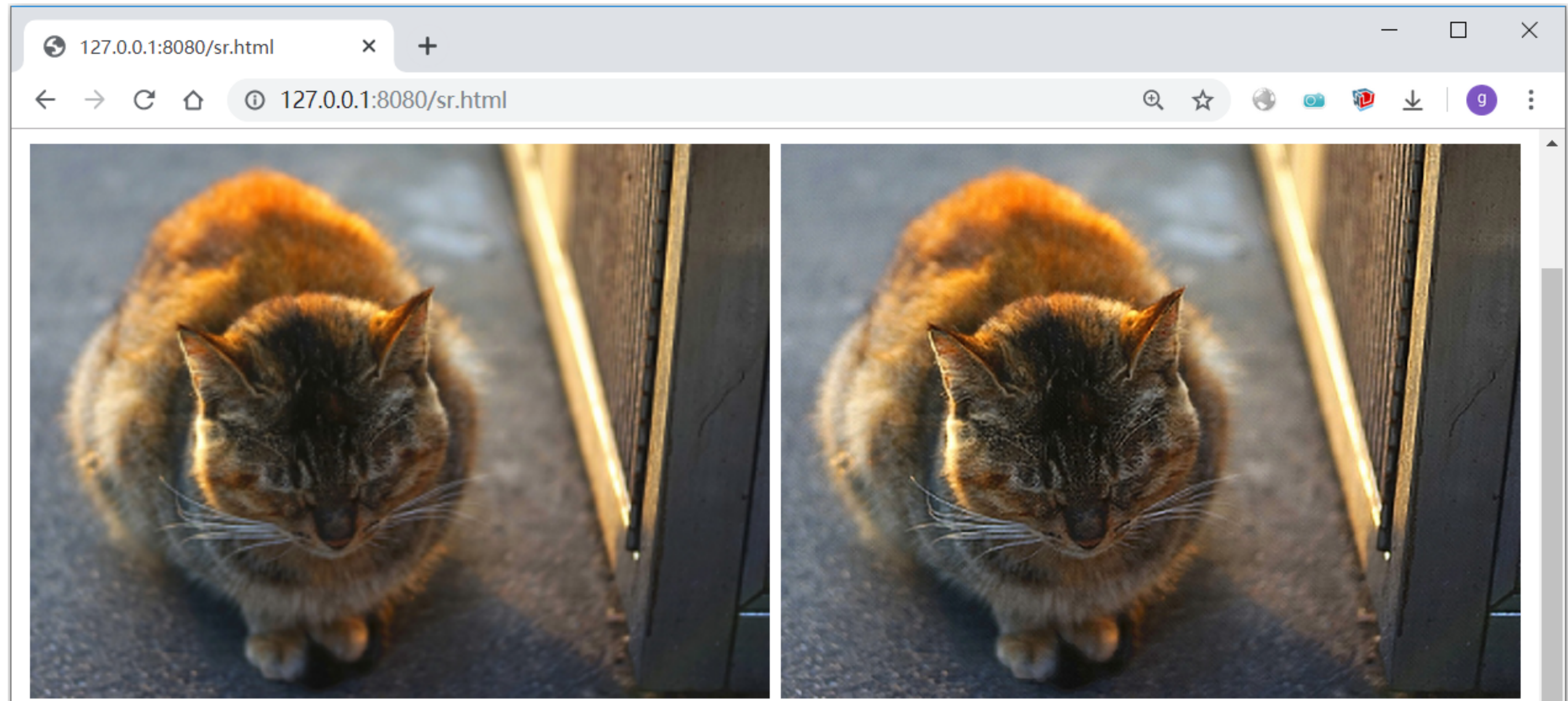
 **Geforce MX250**
NVIDIA



 **UHD Graphics 620**

WebGL视频超分探索

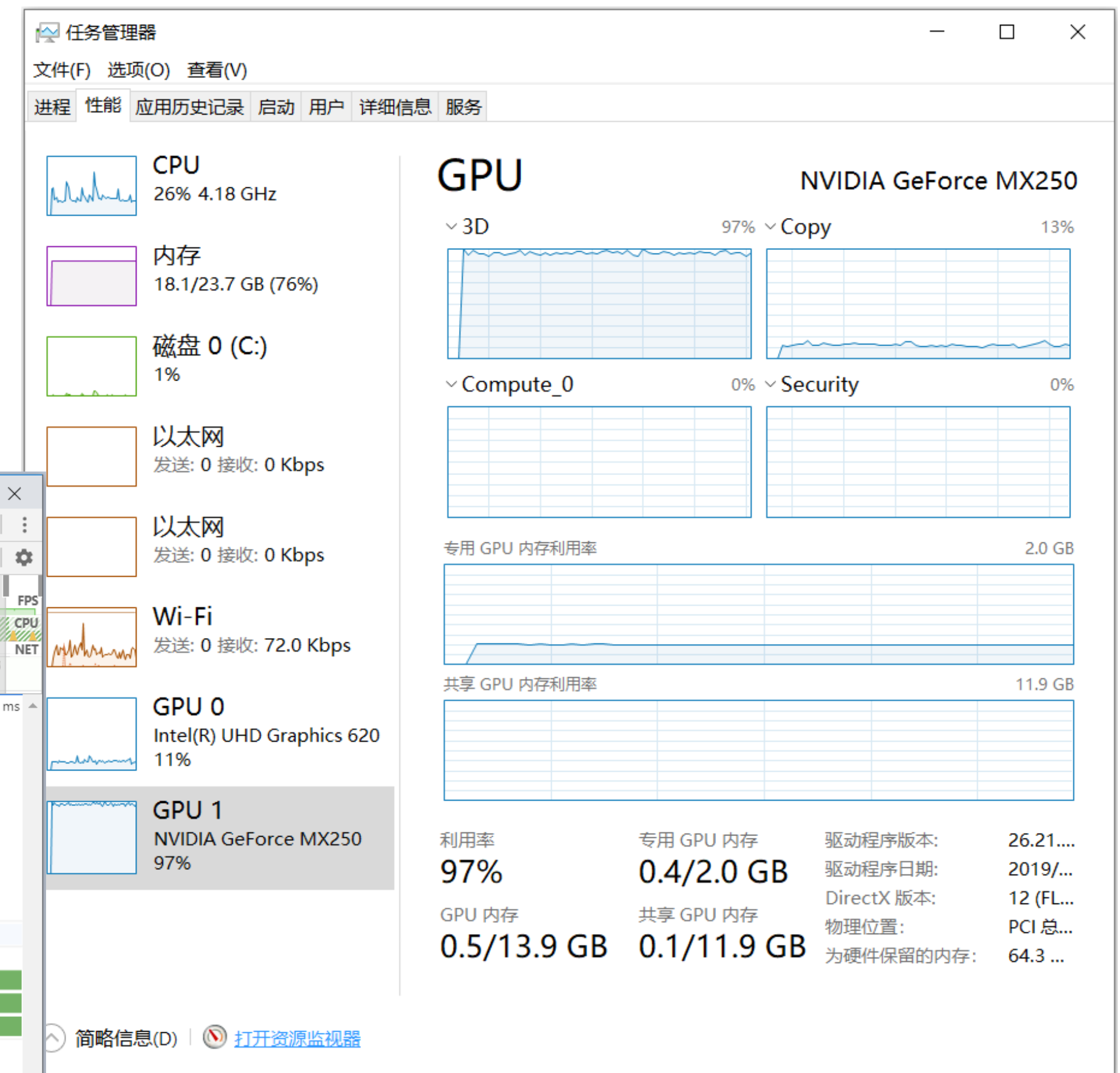
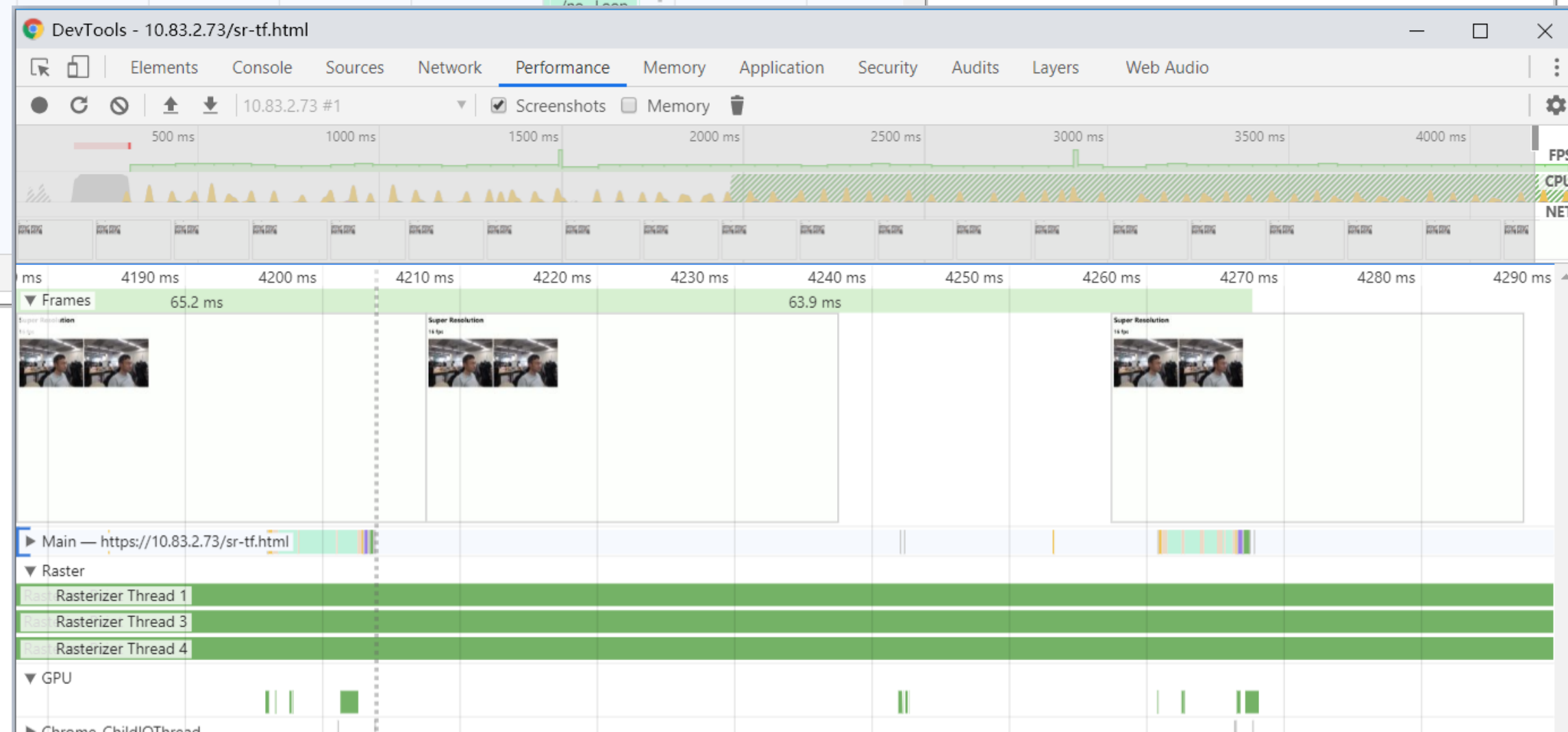
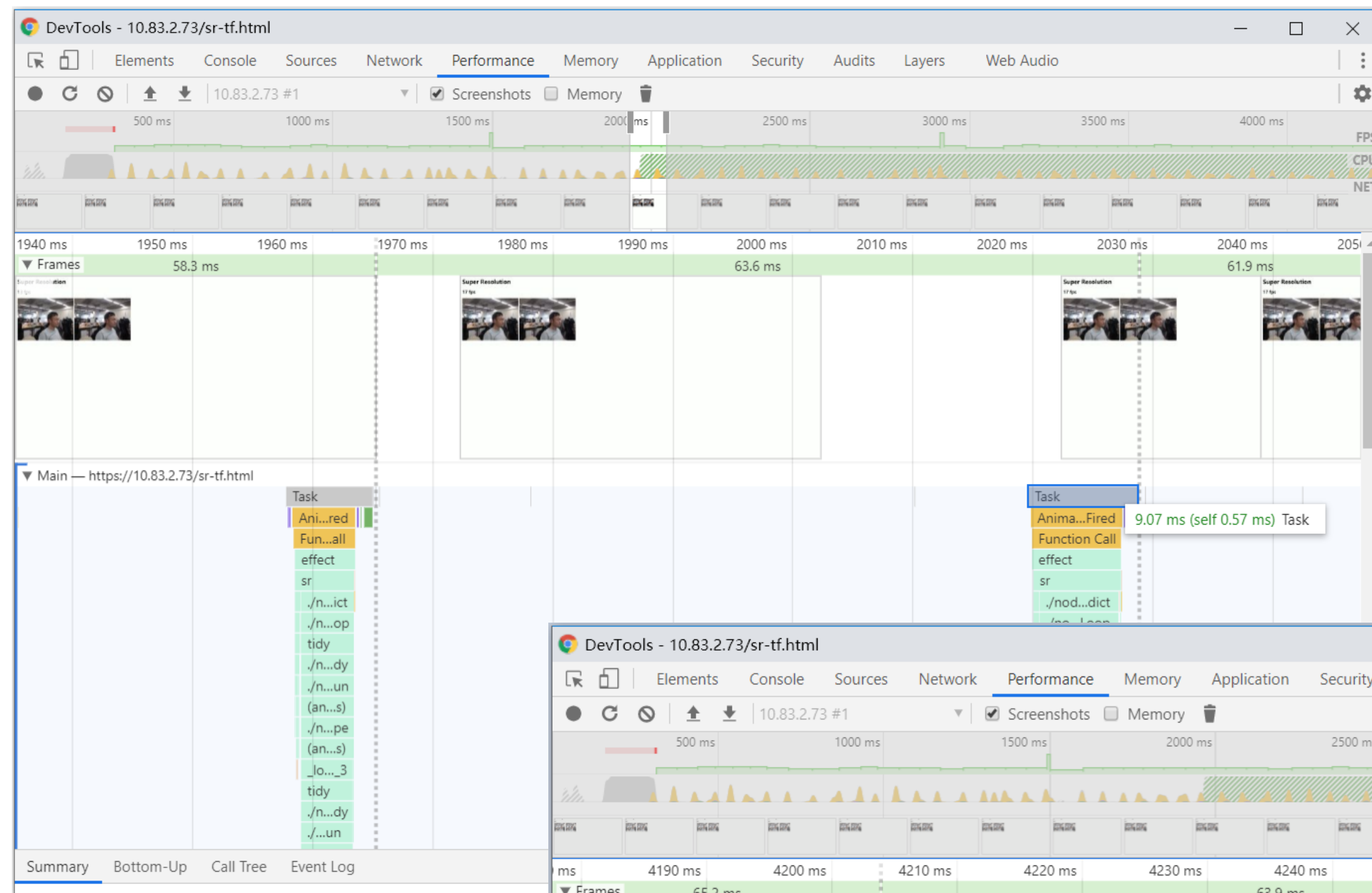
图像效果(320 x 240)



 [视频演示效果](#)

WebGL视频超分探索

基于定制tfjs的视频超分资源开销



Web平台深度学习展望

Web Neural Network API

Draft Community Group Report, 15 October 2019

This version:

<https://webmachinelearning.github.io/webnn/>

Issue Tracking:

[GitHub](#)

Editor:

Ningxin Hu ([Intel Corporation](#))

Explainer:

[explainer.md](#)

Copyright © 2019 the Contributors to the Web Neural Network API Specification, published by the [Machine Learning for the Web Community Group](#) under the [W3C Community Contributor License Agreement \(CLA\)](#). A human-readable [summary](#) is available.

Abstract

This document describes a dedicated low-level API for neural network inference hardware acceleration.

Status of this document

This specification was published by the [Machine Learning for the Web Community Group](#). It is not a W3C Standard nor is it on the W3C Standards Track. Please note that under the [W3C Community Contributor License Agreement \(CLA\)](#) there is a limited opt-out and other conditions apply. Learn more about [W3C Community and Business Groups](#).

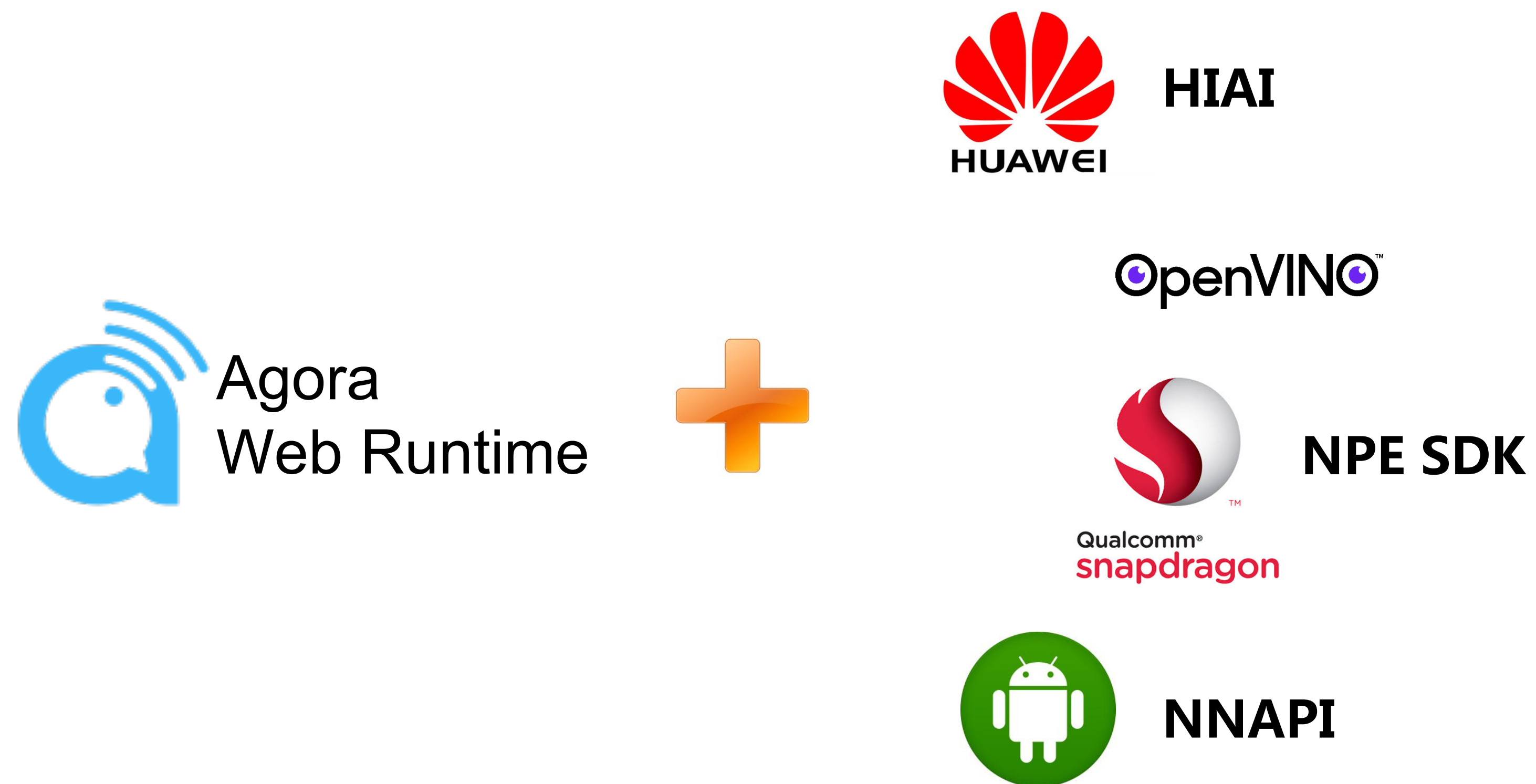
W3C机器学习社区组正在推动的Proposal之一

目标：定义一套硬件加速实现的神经网络专用API

内容：<https://webmachinelearning.github.io/webnn/>



Web平台深度学习展望

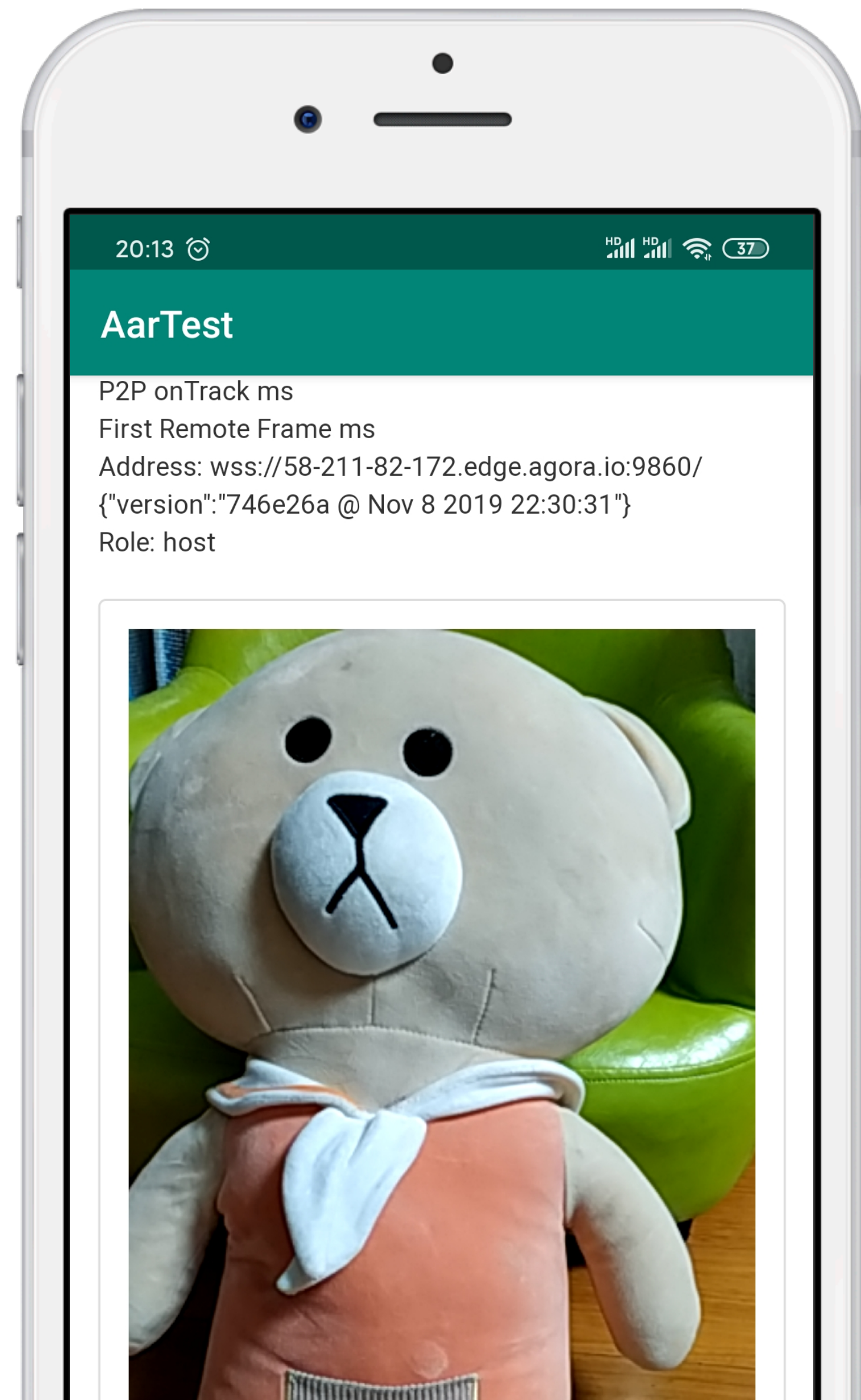


通过定制的Web Runtime将NPU、DSP等硬件能力提供给Web开发者

Web平台深度学习展望

```
1 package com.example.aarTest;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import io.agora.WebView;
7
8 public class MainActivity extends AppCompatActivity {
9
10     private WebView webView;
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         webView = findViewById(R.id.webview);
18         webView.loadUrl("https://webdemo.agora.io/premium_rtc_test_3.0/");
19     }
20 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".MainActivity">
8
9     <io.agora.WebView
10         android:id="@+id/webview"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent" />
13
14 </androidx.constraintlayout.widget.ConstraintLayout>
```



总结

1. WebGL是Web实时视频处理的有效手段
2. 避免使用JS对视频数据进行处理
3. 了解浏览器原理避免低效的WebGL操作
4. 通用处理框架可依据应用场景进行特殊优化
5. 部分浏览器暂时无法满足的需求可通过定制Web引擎实现



关注 QCon 公众号

收获国内外一线大厂实践 与技术大咖同行成长

✓ 演讲视频 ✓ 干货整理 ✓ 大咖采访 ✓ 行业趋势





扫码查看

60 天学透算法与数据结构

600+ 名企内推通道向你打开！

✓ 大厂内推 ✓ 实战演练 ✓ 闭环体系 ✓ 社群连接

THANKS

