Project FoxEye

Bring Modern Image Processing and Computer Vision Technologies to the Web

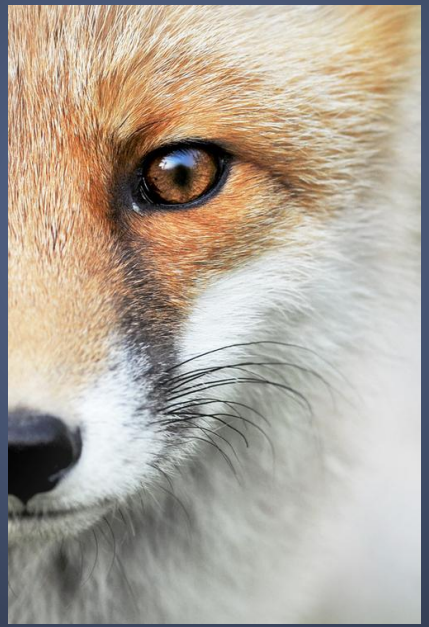
Chia-hung Tai(ctai),
TPE Multimedia,
Mozilla

Why did I start this project?

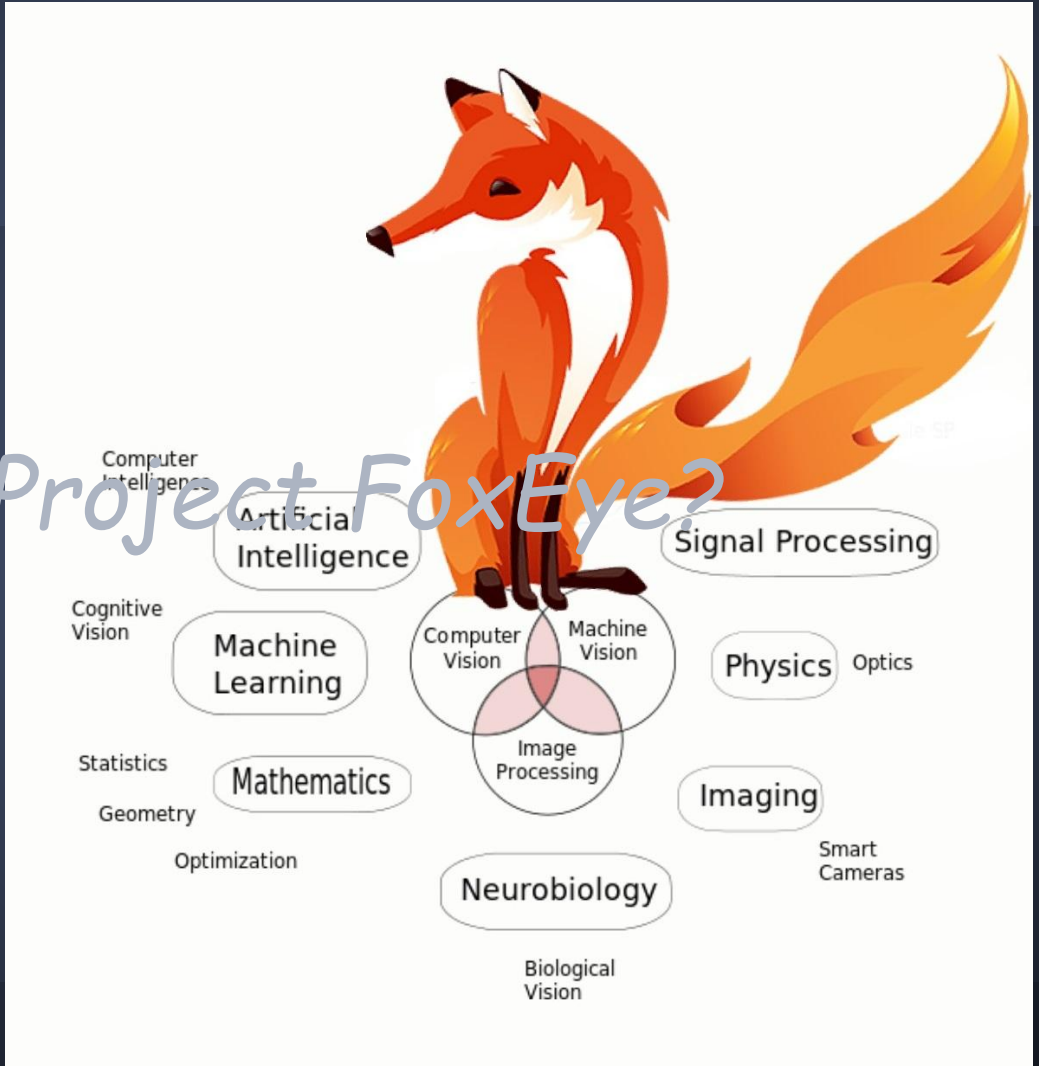
Dyson brings the amazing machine vision possible only in human……

Is it possible to bring machine computer vision technology to the Web?

So, the Project FoxEye came to the world.



What is Project FoxEye?



Let's see some demos!

The open source projects used in the demos

- OpenCV: BSD license, tons of computer vision algorithms library.
- Tesseract-OCR: Apache License v2.0, the most accurate open source Optical Character Recognition engine available.

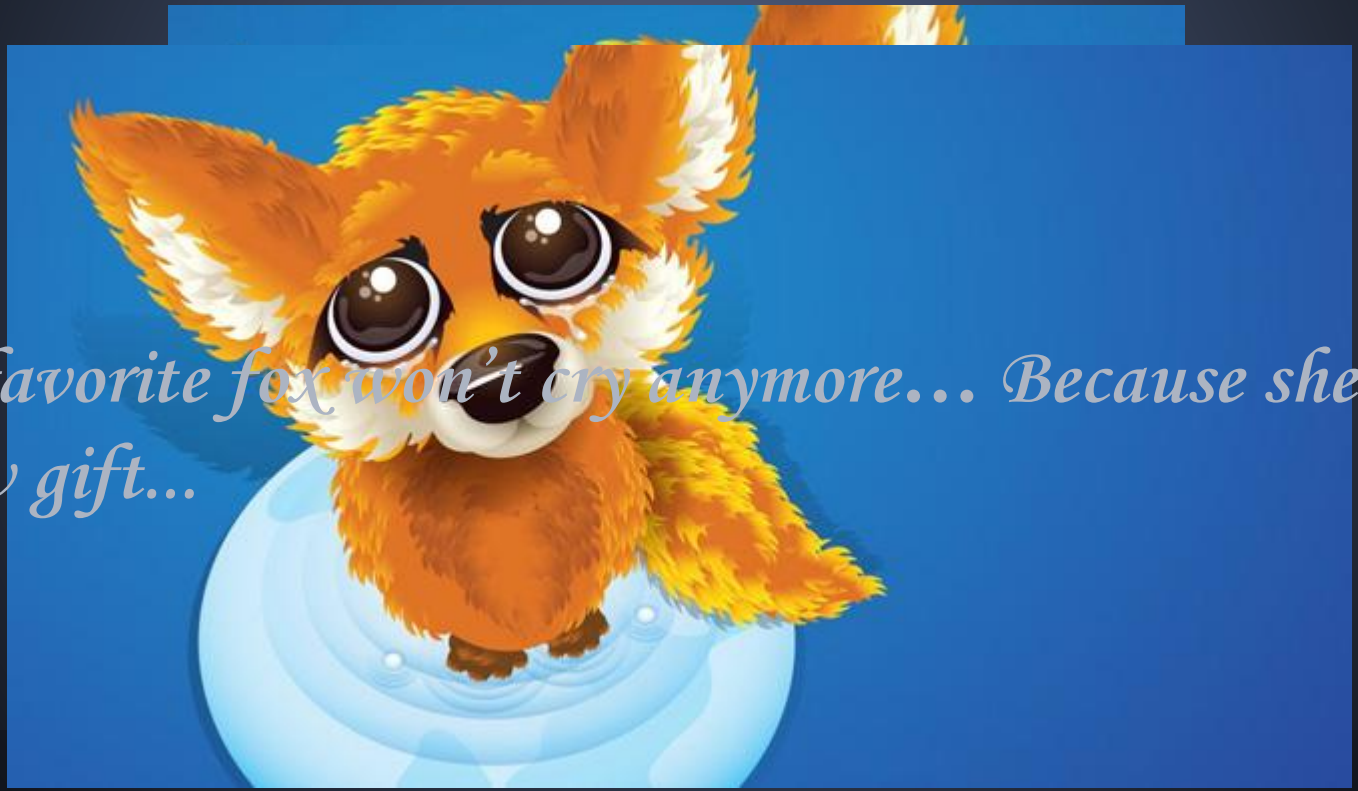
Demos

- FaceTracking On Flame/Browser
  - Use getUserMedia to get the MediaStream.
  - Pass the frame of MediaStream to OpenCV face tracker module.
  - Draw a rectangle around the detected face.

Demos

- Text Recognition On Browser
  - Pass the frame to OpenCV text detection module.
  - Pass detected result to tesseract-ocr
  - Return recognized text back to JavaScript context.

She can read natural language!



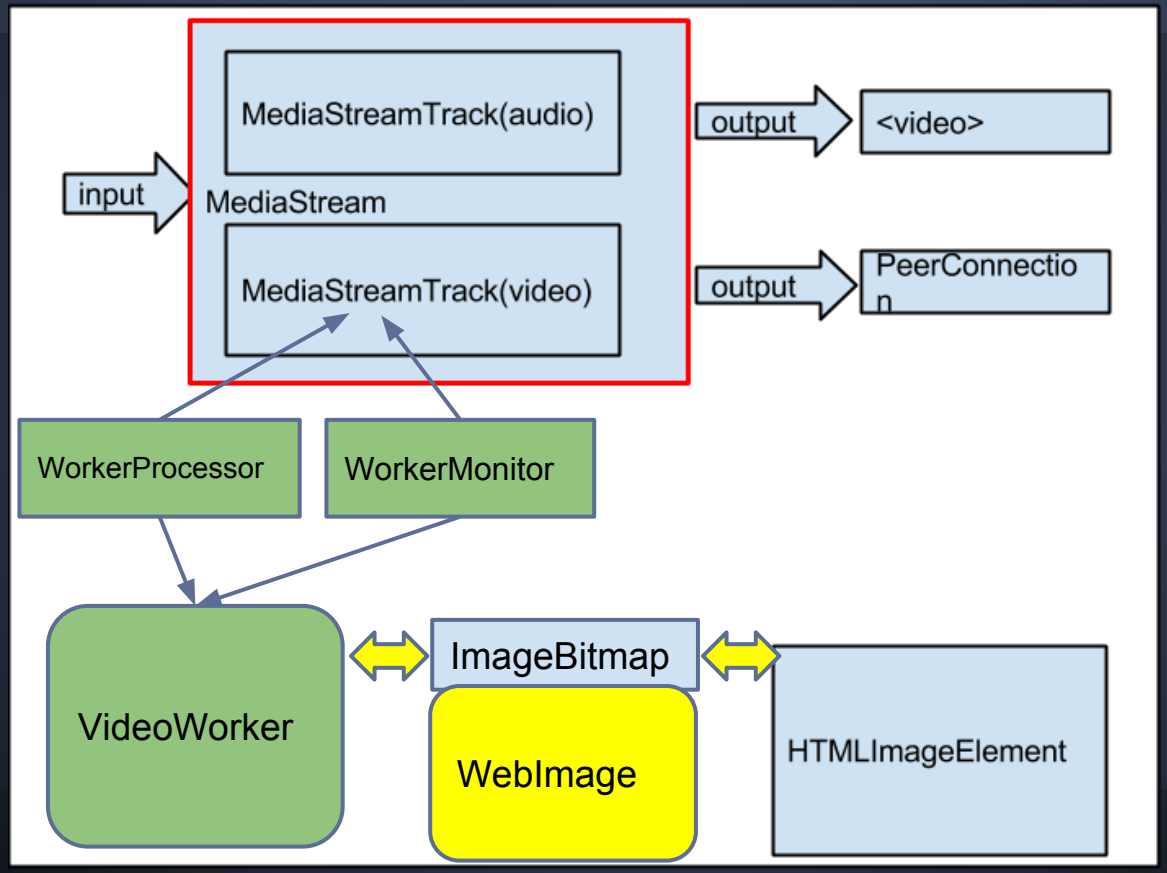
Our favorite fox won't cry anymore… Because she has a new gift…

So, how it works?

How it works

- Provide an image processing and comupter vision library, WebImage, for some features.
- Associate MediaStreamTrack with Workers

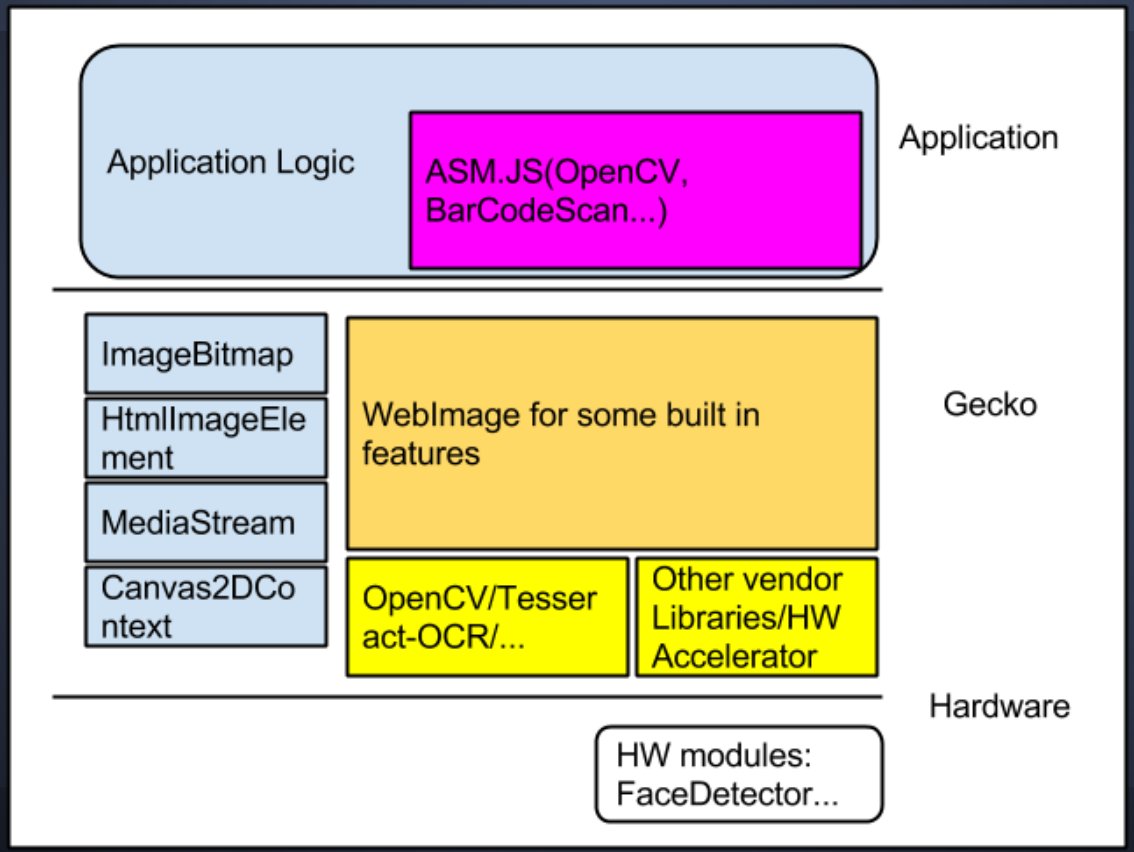
How it works (Cont'd)



WebImage

- Reduce the WebImage scope as small as we can-> use asm.js as much as we can.
- Some image processing/computer vision features in asm.js might run poorly in B2G case.
  - Will do the comparison and analysis later.
- For some HW accelerator case...like face detection in B2G
- Can be cross platforms through asm.js

WebImage(Cont'd)



API Draft

```
partial interface MediaStreamTrack : EventTarget {
// WorkerMonitor is for the case of just frames analysis without any modification.
// 'parameters' cloned and passed in VideoProcessEvent
  void addWorkerMonitor(Worker worker, object parameters);
  void removeWorkerMonitor(Worker worker, object parameters);

// WorkerProcessor is for the case of modifying frames must be generated in real time.
// 'parameters' cloned and passed in VideoProcessEvent
  MediaStreamTrack addWorkerProcessor(Worker worker, object parameters);
  void removeWorkerProcessor(Worker worker, object parameters);
};

interface VideoWorkerGlobalScope : DedicatedWorkerGlobalScope {
  attribute EventHandler onvideoprocess;
};

interface VideoProcessEvent: Event {
  readonly attribute DOMString   trackId;
  readonly attribute double      playbackTime;
  readonly attribute ImageBitmap inputFrame;
           attribute ImageBitmap outputFrame;
  readonly attribute object      parameters;
};
```

Sample code(Main JS)

Main javascript file:

```
1  var myMediaStream;
2  navigator.getUserMedia({video:true, audio:false}, function(localMediaStream) {
3    myMediaStream = localMediaStream;
4    var videoTracks = myMediaStream.getVideoTracks();
5    var track = videoTracks[0];
6    var myWorker = new Worker("textRecognition.js");
7    track.addWorkerMonitor(myWorker));
8    myWorker.onmessage = function (oEvent) {
9      console.log("Worker recognized: " + oEvent.data);
10   };
11   var elem = document.getElementById('videoelem');
12   elem.mozSrcObject = dest.stream;
13   elem.play();
14 }, null);
```

Sample code(worker)

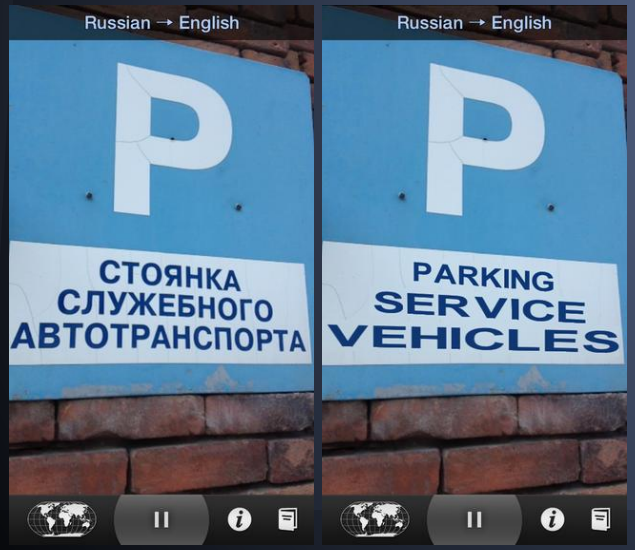
```
var textDetector = WebImage.createTextDetector(img.width, img.height)

onvideoprocess = function (event) {
  var img = event.inputFrame;
  // Do text recognition.
  // We might use built-in detection function or OpenCV in asm.js
  var words= textDetector.findText(img);
  var recognizedText;
  for (var ix = 0; ix < words.length; ix++) {
    recognizedText = recognizedText  + words[ix] + " ";
  }
  postMessage(recognizedText);
}
```

Unlimited Potentials…

Text recognition related use cases

Use cases(Word Lens, Waygo)

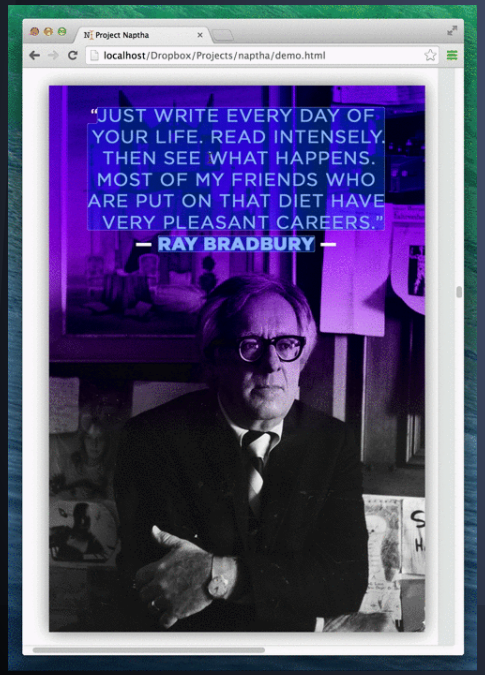
On-fly offline translator with your camera

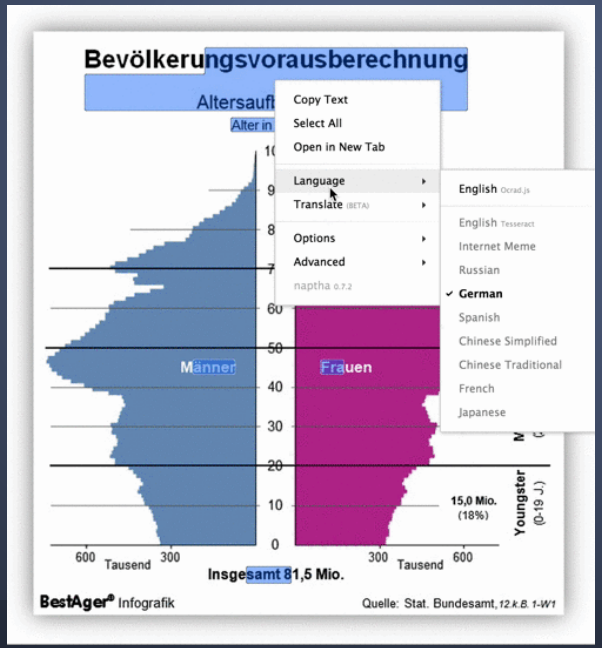


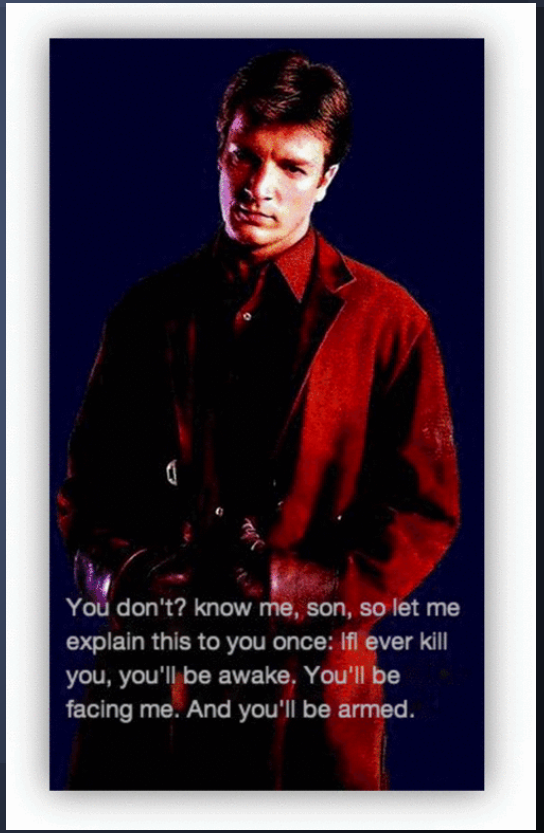
Use cases(Amazon FireFly)

Use cases(Text selection in Image, browser)

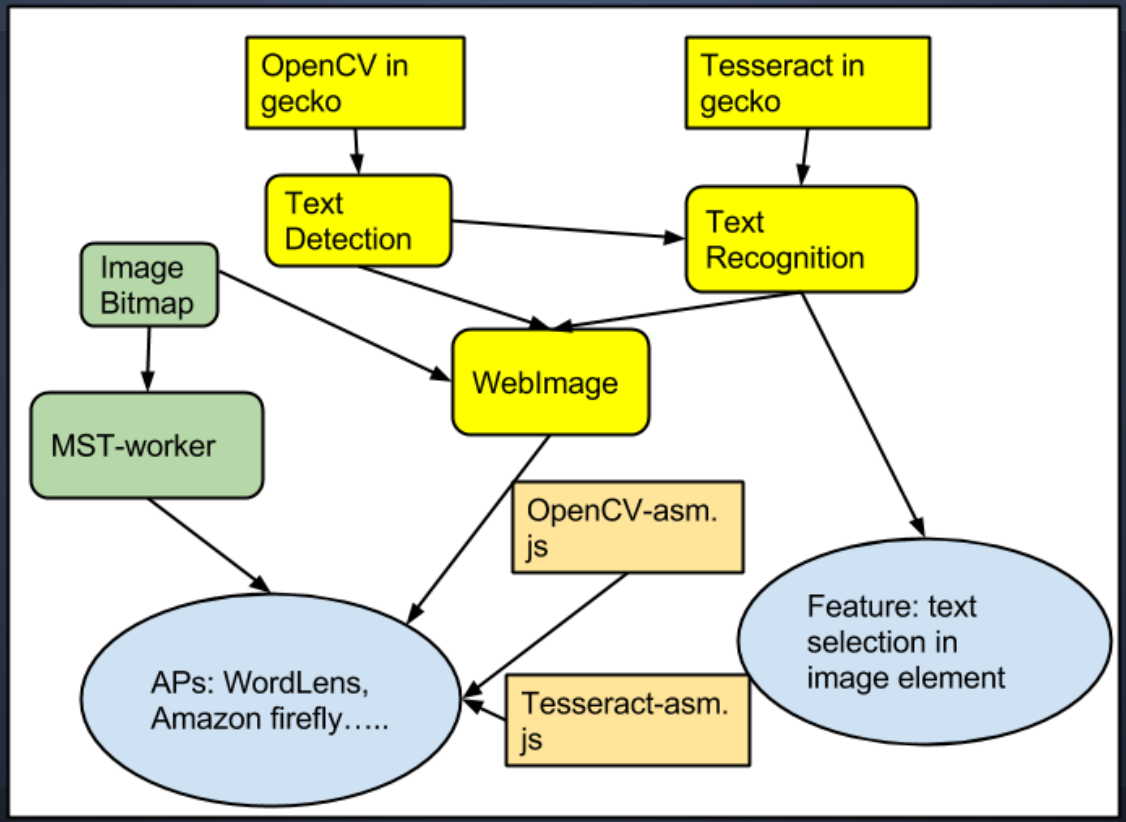
http://projectnaptha.com/







Task dependency(Text recognition)



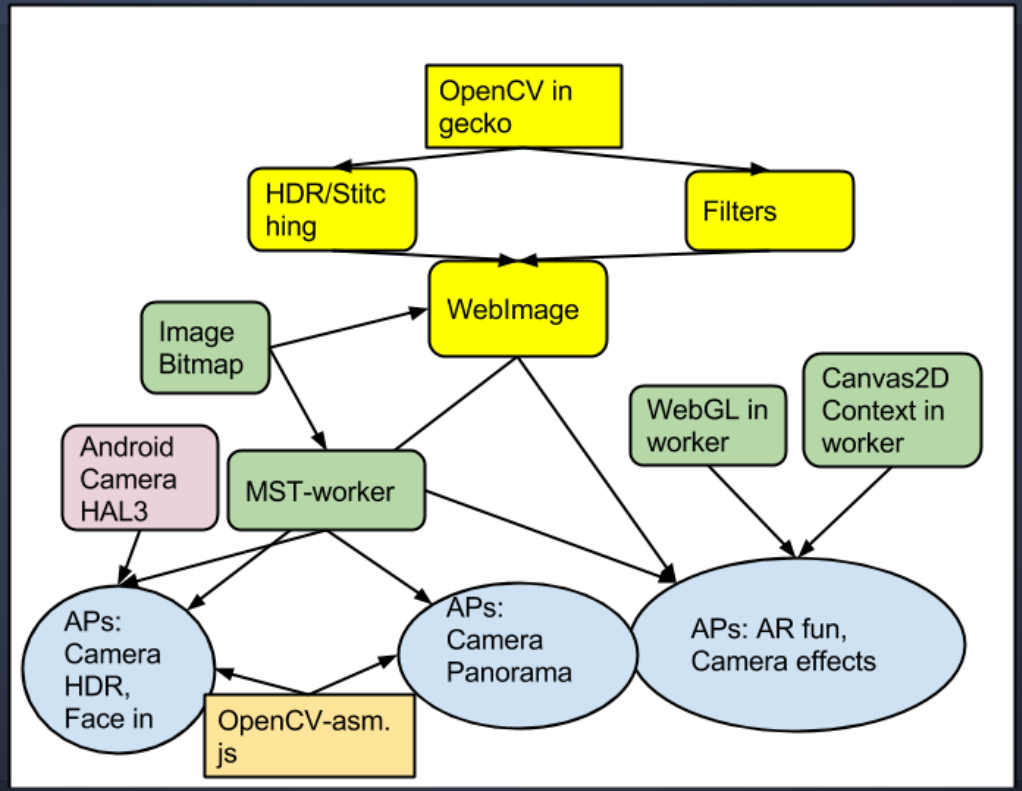
Camera effects related use cases

Use cases(Augmented Reality)

Use cases(Face in)

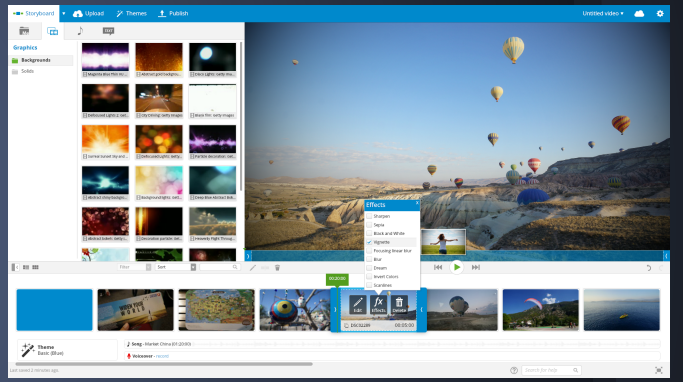
Use cases(Camera Panorama, HDR)

Task Dependency(Camera Effects)



Video editor related use cases

Use cases(VideoEditor)

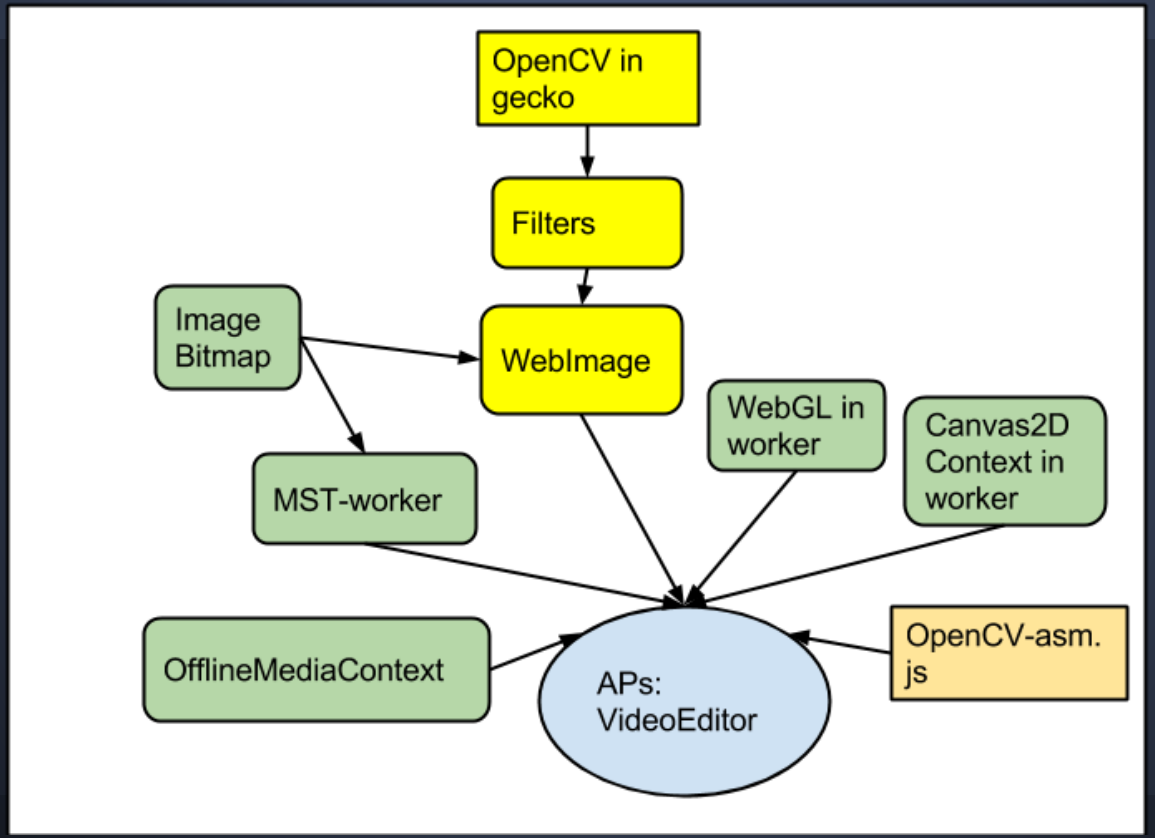


WeVideo, an online Flash based video editor. (https://www.wevideo.com)

Needed features:

- Operations on media
- Effects on media
- Transition effects
- Encode to local files

Task Dependency(Video Editor)



Any possible solution…

Comparison: Canvas2DContext

```
// Compute and display the next frame
    this.renderFrame = function() {
        // Acquire a video frame from the video element
        this.ctx.drawImage(this.video, 0, 0, this.video.videoWidth,
                           this.video.videoHeight,0,0,this.width, this.height);
        var data = this.ctx.getImageData(0, 0, this.width, this.height);
        // Apply image effect
        this.effect.filter(data,this.effect.defaultValues);
        // Render to viewport
        this.viewport.putImageData(data, 0, 0);
    return;
    };
```

Drawbacks(Canvas2DContext)

- It is polling mechanism, you can’t gurantee frame by frame processing
- It use ImageData. (Need to convert in YUV source)
- No offline processing. (Huge problem in video editor case)

Comparison: node-opencv

- It is a native OpenCV binding for node.js.

```
cv.readImage(filename, function(err, mat){
  mat.convertGrayscale()
  mat.canny(5, 300)
  mat.houghLinesP()
})

var s = new cv.ImageStream()
s.on('data', function(matrix){
   matrix.detectObject(haar_cascade_xml, opts, function(err, matches){})
})
ardrone.createPngStream().pipe(s);
```

Drawbacks(node-opencv)

- Need pre-installed OpenCV.
- Node js is for server side JavaScript.
- Hard to integrate with HTMLElement.

Plan

- MST-worker, ImageBitmap for YUV, OfflineMediaContext
- Compare asm.js with native version in B2G.
  - OpenCV in Gecko?
  - Tesseract-OCR in Gecko?
- Module in WebImage(depend on sub-project and the performance consideration)
  - Text detection, text recognition
  - Face detection, face recognition
  - Filters
  - Stitching
  - HDR
  - …

Conclusion

At least 5 potential sub-projects:

- Photo/Video editing tools
- Camera Effect
- Face recognition
- Text recognition(copy/paste/search/translate text in image)
- Shopping Application to integrate with Yahoo Services, like the APP "Amazon".

It might be a chance to build some unique features on Firefox OS/browser through this project.

See https://wiki.mozilla.org/Project_FoxEye and bug 1100203 for more information.

The most important thing...

A lot of un-existing yet use cases are waiting for us to invent…..

Ultimate goal(Smart WebPhone-TBD)

- **Seamlessly connect real-world items to the Web.**
- And fully integrate with web services
  - Yahoo/Mozilla web services, like Yahoo Search/Movie/Shopping/Weather…
  - Utilize location service(Ex: Scan a poster and order near-by movie ticket).
  - Melt in your daily life including eat, clothing, live in the conduct.
- And fully integrate with TV project:
  - Firefox Phone can be a TV remote controller.
  - Integrate with electronic program guide.
  - Firefox OS TV can record TV shows in an easy way.

Q&A

To all Mozillian:

I've got faith to believe.
I can do anything.

From Russel Watson-
Faith of the Heart
(Enterprise Theme)

Let's rock the world!