



UNIVERSIDAD AUTONOMA DE NUEVO LEON
FACULTAD DE INGENIERIA MECANICA Y ELECTRICA



VISION COMPUTACIONAL
DETECCIÓN DE FUEGO MEDIANTE VIDEO

NOMBRE: JESUS ISRAEL PERALES MARTINEZ
MATRICULA: 1443065

El proyecto se llama Fuego, consiste en detectar el fuego o llamas desde la cámara web mediante el navegador, para esto yo utilice javascript en lugar de python y aunque el procesamiento es algo lento, es funcional.

Sinopsis

Los detectores de humo y detectores de calor actualmente proporcionan la primera línea de defensa contra incendios de estructuras catastróficas. Sin embargo, estas tecnologías no son efectivos en grandes espacios abiertos, como túneles, pasillos, arenas, y en el aire libre. Por otra parte, un sistema de detección de incendios a base de vídeo no tiene ninguna de estas deficiencias. Además, una sola cámara puede cubrir un área mucho más grande que un único detector de humo. Sin embargo, las tecnologías actuales de detección de incendios a base de vídeo se basan en buena calidad de vídeo para que funcione correctamente. Por lo tanto, el objetivo de este proyecto es derivar un algoritmo robusto para la detección de fuego y humo en los vídeos de baja calidad en tiempo real.

La sinopsis es obtenida directamente de los proyectos de ejemplo para seleccionar para nuestro proyecto.

<http://vision.ucsd.edu/project/fire-detection-video>

En primera instancia se comenzó por una simple imagen con fuego



Para poder trabajar con la imagen teníamos que cargarla en nuestro HTML y después utilizar el nuevo elemento canvas de HTML5 que nos permite la manipulación de imágenes pixel por pixel.

Para definir si se ha encontrado fuego se utilizó un método muy sencillo que se basa en un diccionario de colores y al estar el color del pixel en el diccionario se toma como que es fuego y se envía a un escuchador donde se busca definir las coordenadas máximas y mínimas en los ejes "X" y "Y" encontrados para posteriormente al finalizar la búsqueda de fuego en cada uno de los píxeles de la imagen original se dibuja el rectángulo utilizando las coordenadas máximas y mínimas anteriormente definidas.

Obteniendo el siguiente resultado:



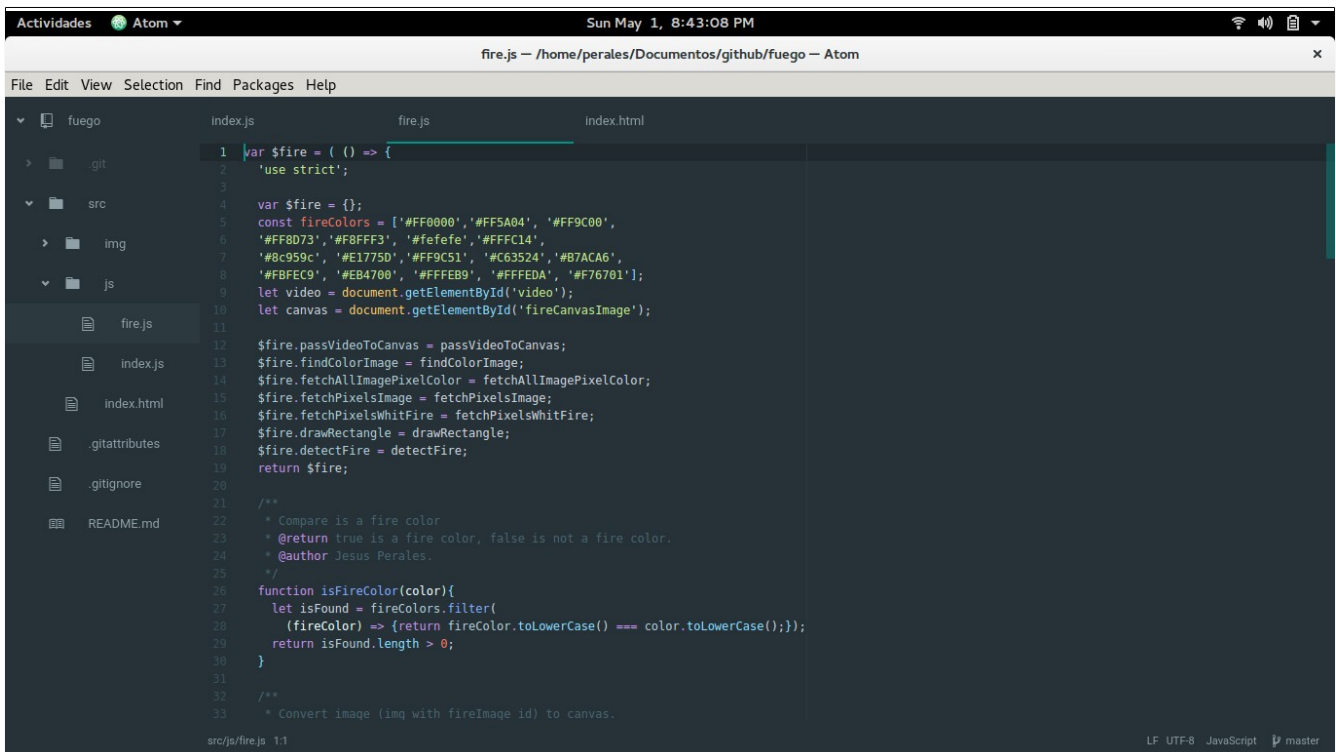
Hay que aclarar que canvas maneja los colores en RGBA y para nuestra comodidad se convierten a colores hexadecimales ya que en el diccionario se manejan en hexadecimales, quizá para optimizar en procesamiento debamos evitar esta transformación y utilizar colores en RGBA directamente.

También se hizo la prueba con otra imagen obteniendo el siguiente resultado:



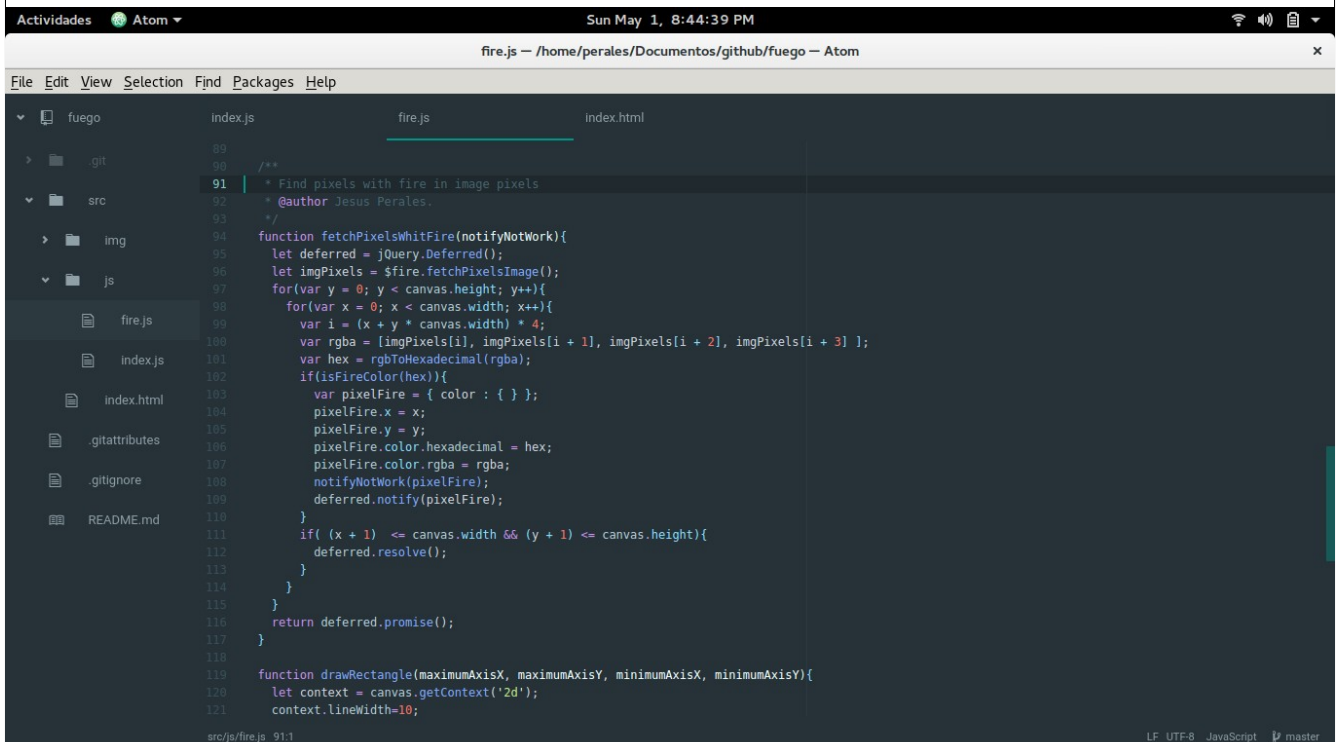
Ahora mostrare parte del código importante

Definición del diccionario de colores y la función que dice si se encuentra o no en el diccionario.



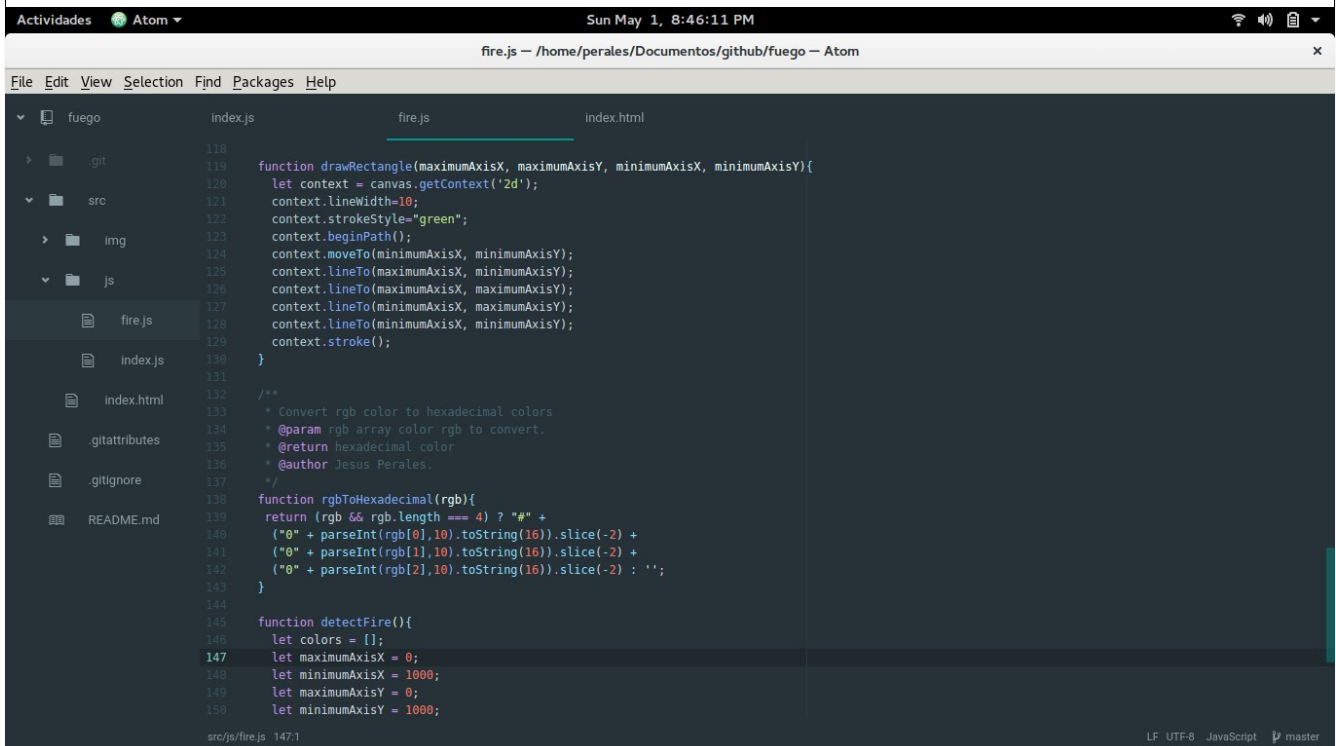
```
1 var $fire = ( () => {
2   'use strict';
3
4   var $fire = {};
5   const fireColors = ['#FF0000', '#FF5A04', '#FF9C00',
6     '#FF8D73', '#F8FF3', '#fefe', '#FFFC14',
7     '#8c959c', '#E1775D', '#FF9C51', '#C63524', '#B7ACA6',
8     '#FBFEC9', '#EB4700', '#FFFE89', '#FFFE8A', '#F76701'];
9   let video = document.getElementById('video');
10  let canvas = document.getElementById('fireCanvasImage');
11
12  $fire.passVideoToCanvas = passVideoToCanvas;
13  $fire.findColorImage = findColorImage;
14  $fire.fetchAllImagePixelColor = fetchAllImagePixelColor;
15  $fire.fetchPixelsImage = fetchPixelsImage;
16  $fire.fetchPixelsWhitFire = fetchPixelsWhitFire;
17  $fire.drawRectangle = drawRectangle;
18  $fire.detectFire = detectFire;
19  return $fire;
20
21  /**
22   * Compare is a fire color
23   * @return true is a fire color, false is not a fire color.
24   * @author Jesus Perales.
25   */
26  function isFireColor(color){
27    let isFound = fireColors.filter(
28      (fireColor) => {return fireColor.toLowerCase() === color.toLowerCase();});
29    return isFound.length > 0;
30  }
31
32  /**
33   * Convert image (img with fireImage id) to canvas.
34  }
35
36  src/js/fire.js 1:1
```

La funcion que busca los pixeles con colores del fuego en todos los pixeles de la imagen



```
91  /**
92   * Find pixels with fire in image pixels
93   * @author Jesus Perales.
94   */
95  function fetchPixelsWhitFire(notifyNotWork){
96    let deferred = jQuery.Deferred();
97    let imgPixels = $fire.fetchPixelsImage();
98    for(var y = 0; y < canvas.height; y++){
99      for(var x = 0; x < canvas.width; x++){
100        var i = (x + y * canvas.width) * 4;
101        var rgba = [imgPixels[i], imgPixels[i + 1], imgPixels[i + 2], imgPixels[i + 3] ];
102        var hex = rgbToHexadecimal(rgba);
103        if(isFireColor(hex)){
104          var pixelFire = { color : { } };
105          pixelFire.x = x;
106          pixelFire.y = y;
107          pixelFire.color.hexadecimal = hex;
108          pixelFire.color.rgba = rgba;
109          notifyNotWork(pixelFire);
110          deferred.notify(pixelFire);
111        }
112        if( (x + 1) <= canvas.width && (y + 1) <= canvas.height){
113          deferred.resolve();
114        }
115      }
116    }
117    return deferred.promise();
118  }
119  function drawRectangle(maximumAxisX, maximumAxisY, minimumAxisX, minimumAxisY){
120    let context = canvas.getContext('2d');
121    context.lineWidth=10;
122  }
123
124  src/js/fire.js 91:1
```


La funcion que dibuja el rectangulo en la imagen:

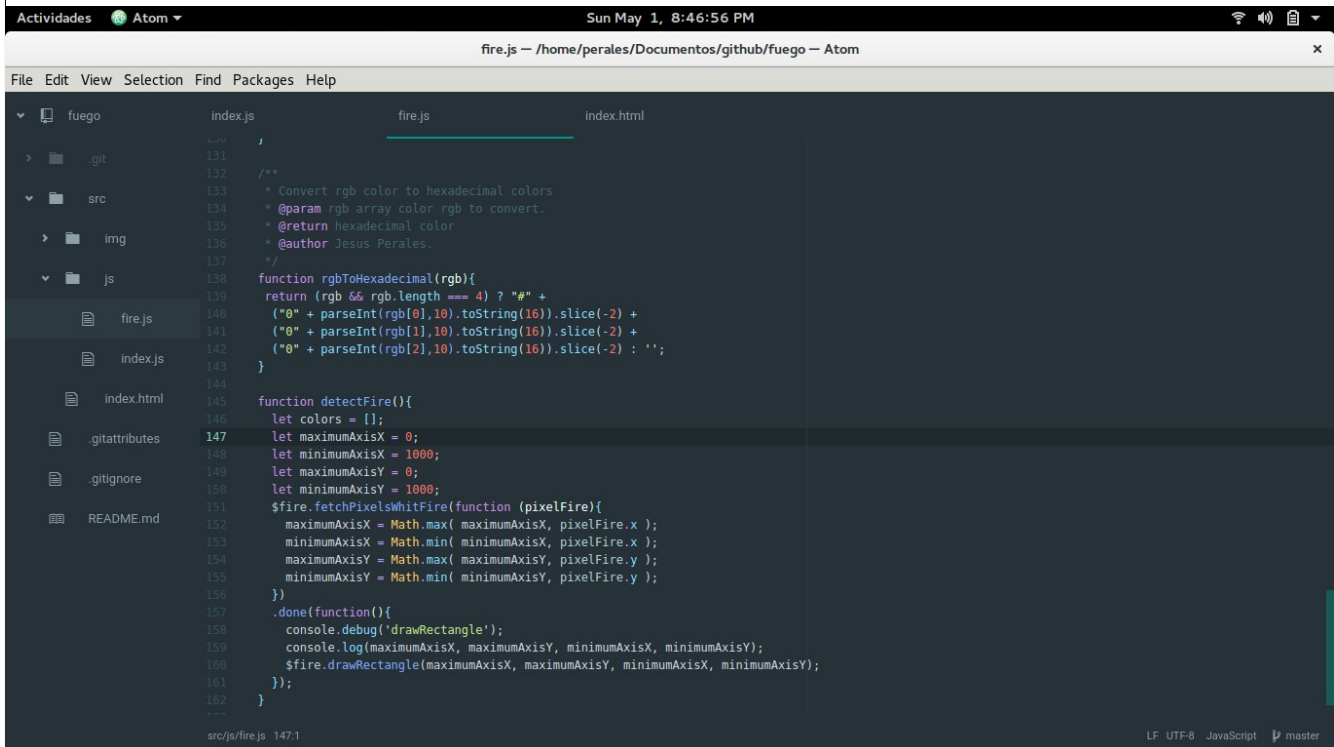


```
118
119 function drawRectangle(maximumAxisX, maximumAxisY, minimumAxisX, minimumAxisY){
120   let context = canvas.getContext('2d');
121   context.lineWidth=10;
122   context.strokeStyle="green";
123   context.beginPath();
124   context.moveTo(minimumAxisX, minimumAxisY);
125   context.lineTo(maximumAxisX, minimumAxisY);
126   context.lineTo(maximumAxisX, maximumAxisY);
127   context.lineTo(minimumAxisX, maximumAxisY);
128   context.lineTo(minimumAxisX, minimumAxisY);
129   context.stroke();
130 }
131
132 /**
133  * Convert rgb color to hexadecimal colors
134  * @param rgb array color rgb to convert.
135  * @return hexadecimal color
136  * @author Jesus Perales.
137  */
138 function rgbToHexadecimal(rgb){
139   return (rgb && rgb.length === 4) ? "#" +
140     ("0" + parseInt(rgb[0],10).toString(16)).slice(-2) +
141     ("0" + parseInt(rgb[1],10).toString(16)).slice(-2) +
142     ("0" + parseInt(rgb[2],10).toString(16)).slice(-2) : '';
143 }
144
145 function detectFire(){
146   let colors = [];
147   let maximumAxisX = 0;
148   let minimumAxisX = 1000;
149   let maximumAxisY = 0;
150   let minimumAxisY = 1000;
```

src/js/fire.js 147:1

LF UTF-8 JavaScript master

En esta captura se observa la funcion que procesa la deteccion del fuego y la funcion que convierte RGBA a hexadecimal.



```
131
132 /**
133  * Convert rgb color to hexadecimal colors
134  * @param rgb array color rgb to convert.
135  * @return hexadecimal color
136  * @author Jesus Perales.
137  */
138 function rgbToHexadecimal(rgb){
139   return (rgb && rgb.length === 4) ? "#" +
140     ("0" + parseInt(rgb[0],10).toString(16)).slice(-2) +
141     ("0" + parseInt(rgb[1],10).toString(16)).slice(-2) +
142     ("0" + parseInt(rgb[2],10).toString(16)).slice(-2) : '';
143 }
144
145 function detectFire(){
146   let colors = [];
147   let maximumAxisX = 0;
148   let minimumAxisX = 1000;
149   let maximumAxisY = 0;
150   let minimumAxisY = 1000;
151   $fire.fetchPixelsWhitFire(function (pixelFire){
152     maximumAxisX = Math.max( maximumAxisX, pixelFire.x );
153     minimumAxisX = Math.min( minimumAxisX, pixelFire.x );
154     maximumAxisY = Math.max( maximumAxisY, pixelFire.y );
155     minimumAxisY = Math.min( minimumAxisY, pixelFire.y );
156   })
157   .done(function(){
158     console.debug('drawRectangle');
159     console.log(maximumAxisX, maximumAxisY, minimumAxisX, minimumAxisY);
160     $fire.drawRectangle(maximumAxisX, maximumAxisY, minimumAxisX, minimumAxisY);
161   });
162 }
```

src/js/fire.js 147:1

LF UTF-8 JavaScript master

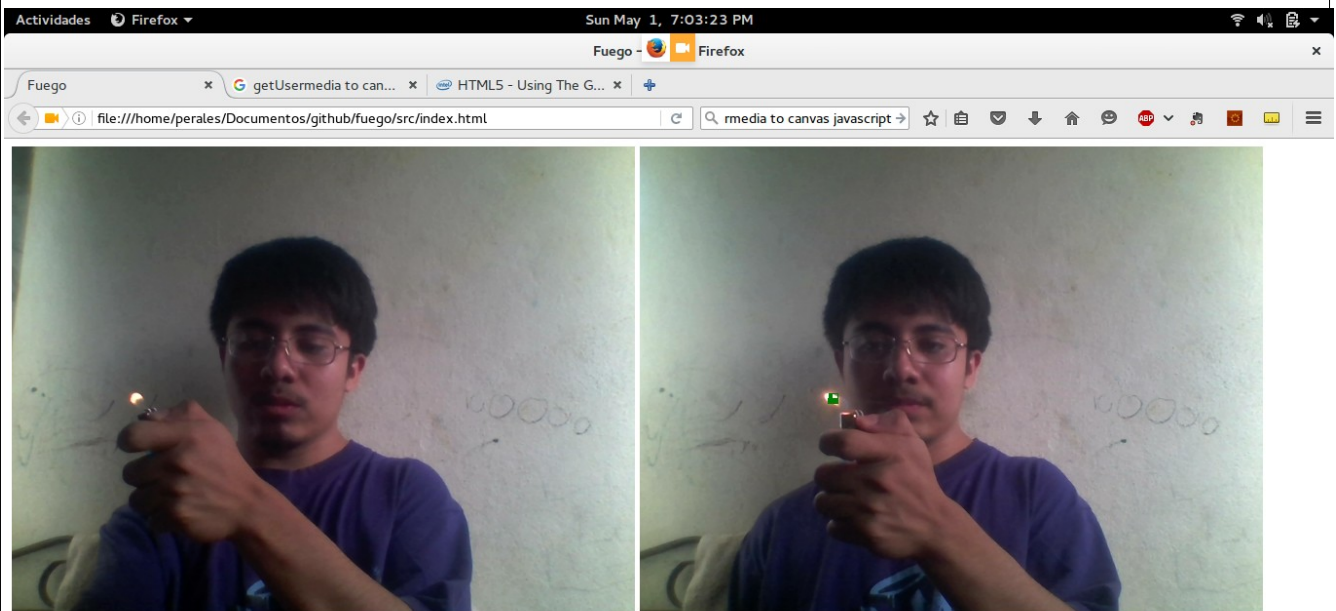
Por el momento el procesamiento en la imagen es lento aunque es funcional y es muy prometedor el poder hacer este tipo de cosas desde el navegador, ya que la mayoría de los teléfonos de ahora cuentan con un navegador web.

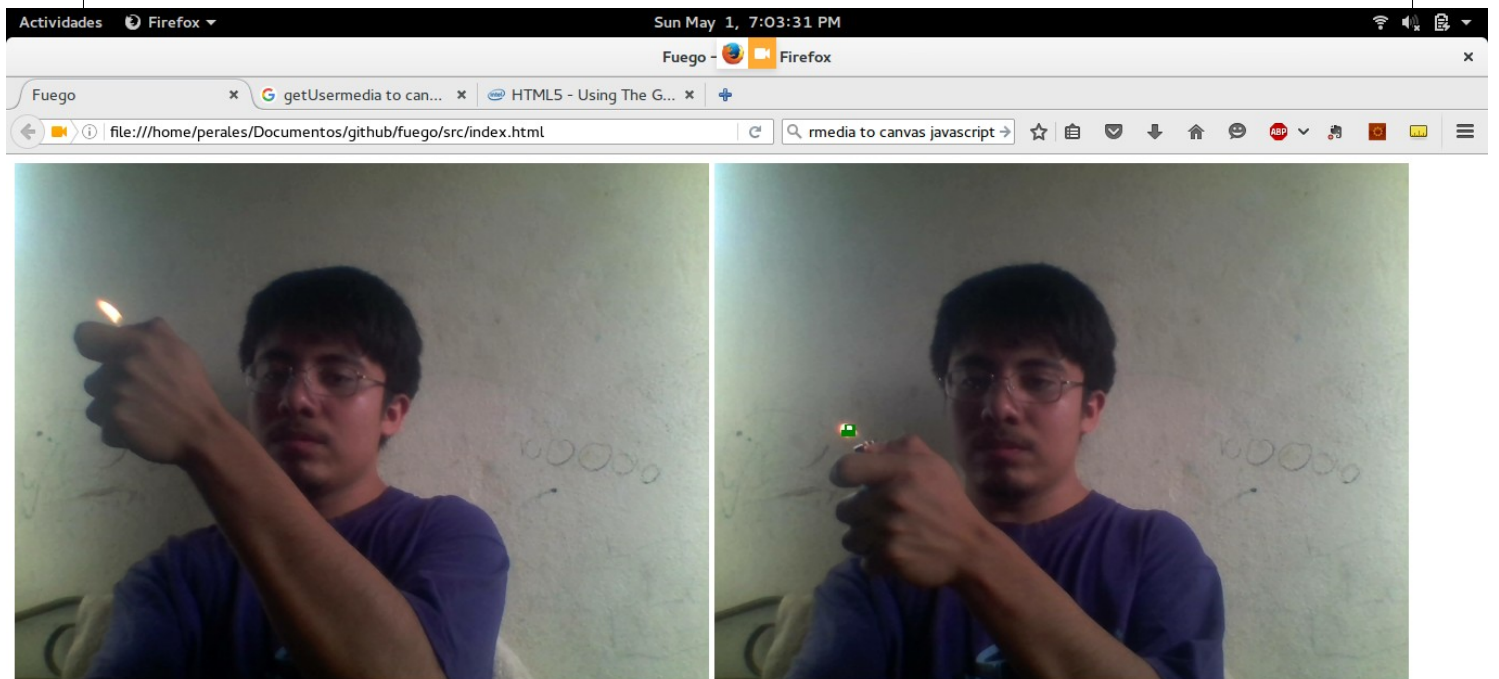
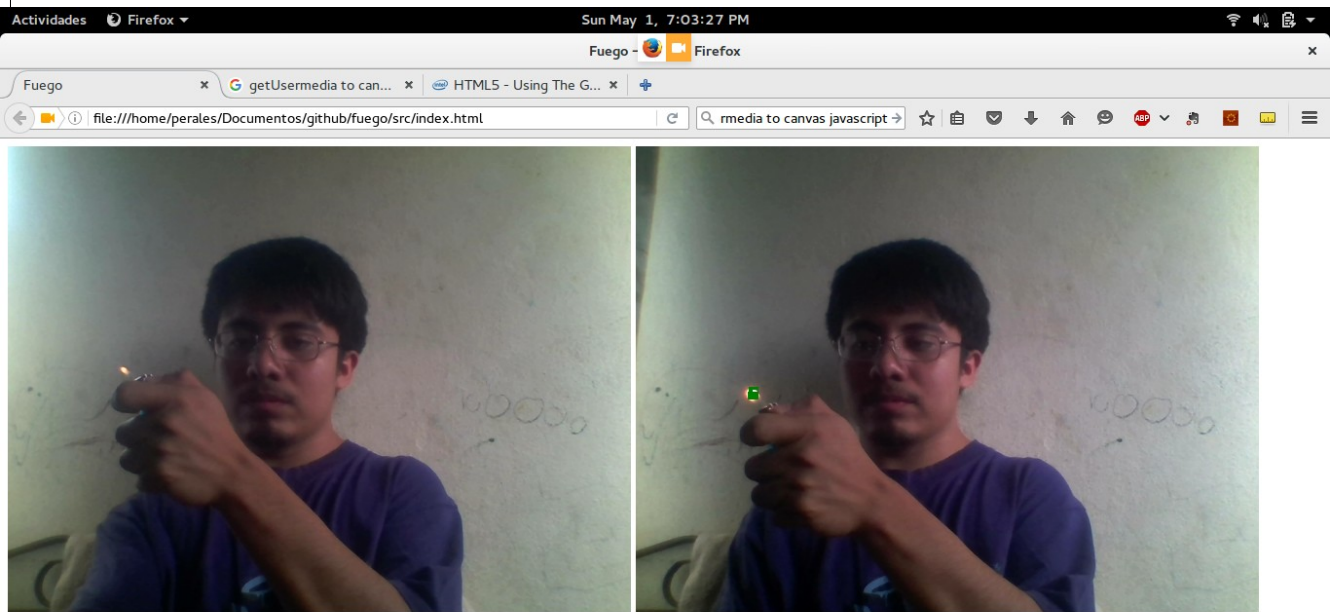
Después de detectar el fuego en una imagen se paso a detectarlo en video.

El proceso es muy similar ya que se seguira trabajando con el objeto canvas de HTML5 solo que ahora en lugar de recibir una imagen y pasarla al procesamiento, se pasan muchas imagenes y se realiza el procesamiento de forma concurrente, solo hay que tener una función que ejecute el procesamiento N veces cada determinado tiempo, debido a que el procesamiento de la imagen es lento se hizo cada 1 segundo y se obtuvieron buenos resultados con la flama de un encendedor.

Hay que detallar que para pasar una imagen a canvas se utilizaba la etiqueta y ahora se utiliza la etiqueta <video> y una función de javascript que obtiene el video de la webcam mediante el objeto getUserMedia y después se refresca el elemento canvas.

Podemos ver el resultado en las siguientes imagenes:





La imagen izquierda es del video continuo, la imagen de la derecha es la captura del canvas cada segundo mas el retardo debido al procesamiento.

Para optimizar el procesamiento de la detección se esta considerando utilizar una nueva especificación de hilos en javascript llamados webworkers el cual ayudara a evitar el atasco en la interfaz de usuario.

También es posible agregar una nueva forma de detectar o definir cuando existe fuego, asi como la clusterización mediante el metodo de kmeans para buscar mas de una llama.

El código de este proyecto se encuentra en Github en la siguiente liga.

<https://github.com/ripper2hl/fuego/>