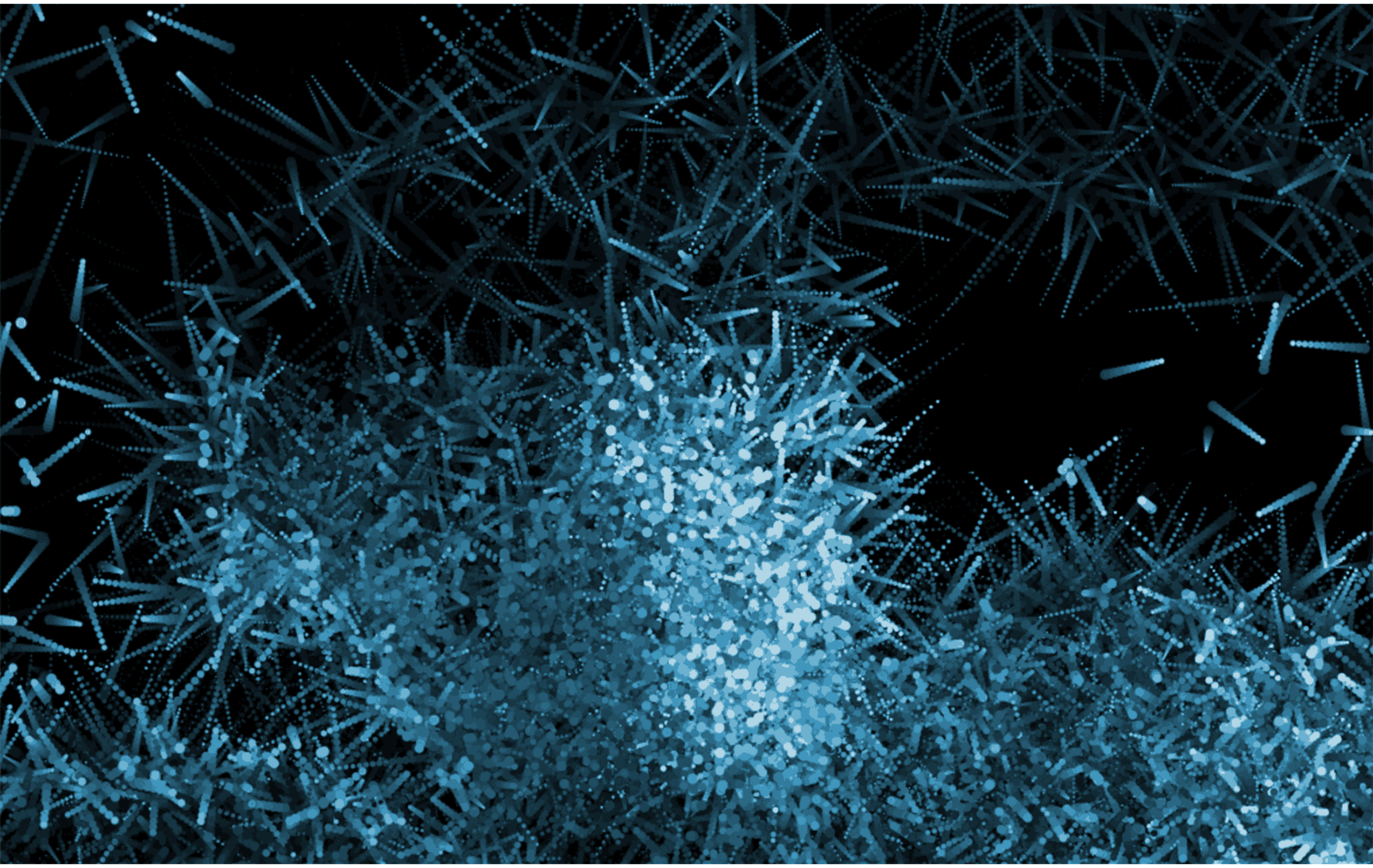

Motion Detection in Javascript

George Gally Follow
Aug 14, 2017 · 7 min read

Making computers see

Part of an ongoing series of learning creative coding in pure javascript. See more here. And get the files here.

There are a bunch of techniques and libraries out there for to detect and play with your camera's motion data. But most often a simple solution is more than you need. And I've found that rolling my own motion detection code gives me better performance, flexibility and understanding. 95% of the time, this simple method is all you're gonna need. So let's get going…

The technique we going to use is simple. We compare the previous frame's colours with the colours of the current frame. If they are above a certain threshold, we can assume there's motion.

Of course there's no need to compare every pixel. This will be too taxing on our computer and often introduces unnecessary noise into the system. So we'll use a similar technique to the previous tutorial Creating a Pixelation Filter.

To get started we need to get your webcam working in the browser. And I'm not going to go into this. There's plenty of tutorials floating around. And it's honestly not that interesting (and kinda verbose). I've created a re-usable Javascript file, so I'll never need to re-write the code or even bother about firing up your webcam… here's the code. Simply include it in your html, as you do with any javascript file, and it will automatically create a *video* element with a default size of 320x240px that is hidden via css.

(Actually could draw it at a smaller size, then scale up our calculations, and get a further performance boost, and I'll show you this in a bit. But let's keep things simple for now).

To draw the video to the screen we just reference the video and use the canvas *drawImage()* function:

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

Now we want to loop through the pixels and get their colour values…

The code is exactly the same as creating a pixelation effect using *getImageData()*, except we're using a live video feed:

```
// setup canvas
var ctx = createCanvas("canvas1");


// sample the colour of every 50 pixels
var sample_size = 50;


function draw(){


  // draw video onto screen
  ctx.drawImage(video, 0, 0, w, h);


  // get the screen's pixels data
  var data = ctx.getImageData(0, 0, w, h).data;


  // loop through rows and columns
  for (var y = 0; y < h; y+= sample_size) {


    for (var x = 0; x < w; x+= sample_size) {


      // the data array is a continuous array of red, blue,
green
      // and alpha values, so each pixel takes up four
values
      // in the array
      var pos = (x + y * w) * 4;

      // get red, blue and green pixel value
      var r = data[pos];
      var g = data[pos+1];
      var b = data[pos+2];


      // draw the pixels as blocks of colours
      ctx.fillStyle = rgb(r, g, b);
      ctx.fillRect(x, y, sample_size, sample_size);


    }
  }


}
```

Testing for motion:

To test if there's motion in the video all we now need to do is store in an array our previous frame's pixel values and compare these to the current values… if the difference is above a certain threshold then there is motion.

We could do a formula for the sum of the RGB values or look atthe brightness, but in most cases really need only to compare the red values…

```
// make an array to hold our old pixel values
var previous_frame = [];
// choose a brightness threshold, if the old pixel values
differs enough then we know there's movement
var threshold = 50;
// sample the colour every 50 pixels
var sample_size = 50;


function draw(){


  ctx.drawImage(video, 0, 0, w, h);
  var data = ctx.getImageData(0, 0, w, h).data;
  ctx.background(0);

  for (var y = 0; y < h; y+= sample_size) {

     for (var x = 0; x < w; x+= sample_size) {


      var pos = (x + y * w) * 4;
      var r = data[pos];
      var g = data[pos+1];
      var b = data[pos+2];


      // first check if it's not the first frame, but
       // seeing of when the previous_frame array
      // is not we empty, and then only draw something if
there's
      // a significant colour difference
      if(previous_frame[pos]
      && Math.abs(previous_frame[pos] - r) > threshold) {
        ctx.fillStyle = rgb(r, g, b);
        ctx.fillRect(x, y, sample_size, sample_size);
      }
      // store these colour values to compare to the next
frame
      previous_frame[pos] = r;


    }

  }


}
```

And there you have it. That’s pretty much the basics of motion detection. Pretty neat and simple.

To my our code easier to read and deal with, let’s quickly put the motion detection code into a function. There’s not much we need to change, except add a motion array to store all the motion points and their colour values, and then return those:

```
function motionDetection(){



  // create an array to store our motion data
  var motion = [];


  ctx.drawImage(video, 0, 0, w, h);
  var data = ctx.getImageData(0, 0, w, h).data;
  ctx.background(0);

  for (var y = 0; y < h; y+= sample_size) {

     for (var x = 0; x < w; x+= sample_size) {


      var pos = (x + y * w) * 4;
      var r = data[pos];
      var g = data[pos+1];
      var b = data[pos+2];


      // first check if it's not the first frame, but
      // seeing of when the previous_frame array
      // is not we empty, and then only draw something if
there's
      // a significant colour difference
      ctx.drawImage(video, 0, 0, w, h);
      var data = ctx.getImageData(0, 0, w, h).data;
      ctx.background(0);

      for (var y = 0; y < h; y+= sample_size) {

        for (var x = 0; x < w; x+= sample_size) {


           var pos = (x + y * w) * 4;
           var r = data[pos];
           var g = data[pos+1];
           var b = data[pos+2];
           // first check if it's not the first frame, but
           // seeing of when the previous_frame array
           // is not we empty, and then only draw something
if
           // a significant colour difference there's
           if(previous_frame[pos]
           && Math.abs(previous_frame[pos] - r) > threshold)
{

          // push the x, y and rgb values into the motion
array
          motion.push({x: x, y: y, r: r, g: g, b: b});

        }


        // store these colour values to compare to the next
frame
        previous_frame[pos] = r;


     }

  }


return motion;


}
```

And then in our draw loop we can just loop through the motion array and do something with the resulting values:

```
function draw(){


   ctx.background(250);
   var motion = motionDetection();
   for (i = 0; i < motion.length; i++) {

      var m = motion[i];
      ctx.fillStyle = rgb(m.r, m.g, m.b);
      ctx.fillEllipse(m.x, m.y, sample_size, sample_size);


    }


}
```

Now let’s work on that performance. As is, it’s pretty fast. But we can make it faster. If you think about it, drawing the video to the whole screen and then reading every 10 or 20 pixels is a bit silly, when we could just make a smaller video input and read those pixels… So let’s do that…

Firstly let's make our video even smaller… We'll use a setup() function which will ensure the video is loaded before we manipulate it… and then we just set the video to a scale of our screen size…

```
var scalefactor = 40;


function setup(){
   video.width = w/scalefactor;
   video.height = h/scalefactor;
}
```

We'd then just need to loop over out video instead of the whole screen, and seeing our video is tiny, we'll get a nice bump in performance. And because we're looking over every pixel of the video now, there's also a tiny reduction in noise. We'd also need to multiply the x and y positions back up by the scalefactor…

```
function motionDetection(){


  var motion = [];


  // draw the video and get its pixels
  ctx.drawImage(video, 0, 0, video.width, video.height);
  var data = ctx.getImageData(0, 0, video.width,
video.height).data;


  // we can now loop over all the pixels of the video
  for (var y = 0; y < video.height; y++) {
    for (var x = 0; x < video.width; x++) {


      var pos = (x + y * video.width) * 4;
      var r = data[pos];
      var g = data[pos+1];
      var b = data[pos+2];


      if(old[pos] && Math.abs(old[pos].red - r) > threshold)
{
        ctx.fillStyle = rgb(r, g, b);
        ctx.fillRect(x * scalefactor, y * scalefactor,
scalefactor, scalefactor);
        // push the x, y and rgb values into the motion
array
        // but multiply the x and y values bck up by
scalefactor
        // to get their actual screen position
        motion.push({x: x * scalefactor, y: y * scalefactor,
r: r, g: g, b: b});


      }
      old[pos] = { red: r, green: g, blue: b};

}
  }

return motion;

}
```

And there you have it. A simple chunk of re-usable motion reactive code. Once you have your motion array, try attaching particles.

Watch out for ambient light, which could add noise and adjust your threshold up and down accordingly.

And easy way for us to make tweaking easier is to add some sliders to adjust our *threshold* …

I like to use bit101's quicksettings: https://github.com/bit101/quicksettings *(Also check out his lab and Coding Math video series for some great creative coding examples and tips)*.

Once you've included the library, you instantiate it like so:

```
var settings = QuickSettings.create();
```

And then add Range sliders for *threshold* and *sample_size*:

```
// addRange(label, min, max, start, increment, callback)
settings.addRange("threshold", 5, 100, 50, 1,
function(value) {
  threshold = value;
});
```

Another technique that's often used, for installations especially, is *background subtraction*. The code is exactly the same. But instead of comparing the previous frame's pixels, you take a snapshot on say a keypress, of your empty room, and then compare that image to your current pixels. Try get it going, and hit me up in the comments or on twitter if you like if you're struggling and I'd be happy to help.

And that’s pretty much all you need for motion detection. Happy coding.

As usual the full code is available on my github: https://github.com/GeorgeGally/creative_coding

You can see previous all my previous creative coding tutorials here.

. . .

And follow me here if you so desire:

https://www.instagram.com/radarboy3000/

https://twitter.com/radarboy_japan

https://www.facebook.com/radarboy3000