Experiment 6

Gaurav Bhagchandani
Roll no. 10
Batch D

Aim: To apply line and edge detection techniques to images

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .container{
            float: right;
            width:600px;
        }
        input[type="text"]{
            width:40px;
        }
        input{
            margin:5px;
            font-size: 15px;
            font-weight: bold;
            font-family: sans-serif;
        }
        .gs{
            margin-top: 10px;
        }
        .thresh{
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <canvas id="canvas" width="800" height="500"></canvas>
    <div class="container">
        <div class="thresh">
            <input id="threshold" type="range" default=100 min=0 max=255
onchange="updateTextInput(this.value);">
            <input type="text" id="textInput" value="100">
            <input id="thresholdapply" value="Threshold" type="button">
        </div>
        <input id="invertbtn" value="Invert" type="button">
        <div class="gs">
            <div>
                <input id="greya" type="range" value=50 min=0 max=255
onchange="updateTextInputGreyA(this.value);">
                <input type="text" id="textGreyA" value="50">
            </div>
            <div>
                <input id="greyb" type="range" value=100 min=0 max=255
onchange="updateTextInputGreyB(this.value);">
```

```
            <input type="text" id="textGreyB" value="100">
        </div>
        <input id="gsb" value="GS With Background" type="button">
        <input id="gswb" value="GS Without Background" type="button">
    </div>
    <input id="smoothing" value="Smoothing" type="button">
    <input id="sharpening" value="Sharpening" type="button">
    <br/>
    <input id="erosion" value="Erosion" type="button">
    <input id="dilation" value="Dilation" type="button">
    <br/>
    <input id="opening" value="Opening" type="button">
    <input id="closing" value="Closing" type="button">
    <br/>
    <input id="fourier" value="DFT" type="button">
    <input id="invfourier" value="IDFT" type="button">
    <br/>
    <input id="horiz" value="Horizontal Line Detection" type="button">
    <input id="vert" value="Vertical Line Detection" type="button">
    <input id="ldiag" value="Left Diagonal Line Detection" type="button">
    <input id="rdiag" value="Right Diagonal Line Detection" type="button">
    <input id="robert" value="Robert's Cross Edge Detection" type="button">
    <input id="sobel" value="Sobel Operator Edge Detection" type="button">
    <input id="prewitt" value="Prewitt Edge Detection" type="button">
</div>
<script>
var img = new Image();
img.src = 'http://localhost:8000/image.jpg';
img.onload = function() {
    draw(this);
};
function updateTextInput(val) {
    document.getElementById('textInput').value=val;
}
function updateTextInputGreyA(val) {
    document.getElementById('textGreyA').value=val;
}
function updateTextInputGreyB(val) {
    document.getElementById('textGreyB').value=val;
}
function addComp(a,b){
    let c = {};
    c.real = a.real+b.real;
    c.imag = a.imag+b.imag;
    return c;
}

function subComp(a,b){
    let c = {};
    c.real = a.real-b.real;
    c.imag = a.imag-b.imag;
    return c;
}
function magComp(a){
    return Math.sqrt(a.real*a.real + a.imag*a.imag);
}
function divComp(a,b){
    let c = {},
        temp = (b.real*b.real + b.imag*b.imag);
    c.real = (a.real*b.real + a.imag*b.imag) / temp;
    c.imag = (a.imag*b.real - a.real*b.imag) / temp;
    return c;
}
function mulComp(a,b){
    let c = {};
    c.real = a.real*b.real - a.imag*b.imag;
    c.imag = a.real*b.imag + b.real*a.imag;
    return c;
}

function doubleuNnk(N,n,k){
    let hold = k*n;
    hold /= N;
    let temp = {},
        theta = -2*Math.PI*hold;
    temp.real = Math.cos(theta);
    temp.imag = Math.sin(theta);
    return temp;
}
function arrToComp(arr){
    arr.forEach((a,i)=>{
        let temp = {};
        temp.real = a;
        temp.imag = 0;
        arr[i] = temp;
    });
    return arr;
}
function TwoDIDFT(arr){
    let temp = [];
    console.log(arr);
    for(let i = 0; i < arr.length; i++){
        temp.push([]);
        for(let j = 0; j < arr[i].length; j++){
            let t = {real:0,imag:0};
            for(let k = 0; k < arr[i].length; k++){
                let o = mulComp(doubleuNnk(arr[i].length,-k,j),arr[i][k]);
                t = addComp(t,o)
            }
            temp[i].push(divComp(t,{real:arr[i].length, imag:0}));
        }
    }
    console.log("INVERSE ROW DONE");
    let result = [];
    for(let i = 0; i < temp.length; i++){
        result.push([]);
        for(let j = 0; j < temp[i].length; j++){
            let t = {real:0,imag:0};
            for(let k = 0; k < temp.length; k++){
                let o = mulComp(doubleuNnk(temp.length,-k,i),temp[k][j])
                t = addComp(t,o)
            }
            result[i].push(divComp(t,{real:temp.length, imag:0}));
        }
    }
    console.log("INVERSE COLUMN DONE");
    for(let i = 0; i < result.length; i++){
        for(let j = 0; j < result[i].length; j++){
            if((i+j) % 2 === 1){
                result[i][j] = mulComp(result[i][j],{real: -1, imag: 0})
            }
        }
    }
    return result;
}
function TwoDDFT(arr){
    for(let i = 0; i < arr.length;i++){
        for(let j = 0; j < arr[i].length; j++){
            if((i+j) % 2 === 1){
                arr[i][j] = -arr[i][j];
            }
        }
    }
    arr.forEach((a)=>{
        arrToComp(a);
    })
    let temp = [];
    let max = 0;
    for(let i = 0; i < arr.length; i++){
        temp.push([]);
        for(let j = 0; j < arr[i].length; j++){
            let t = {real:0,imag:0};
            for(let k = 0; k < arr[i].length; k++){
                let o = mulComp(doubleuNnk(arr[i].length,k,j),arr[i][k]);
                t = addComp(t,o)
            }
            temp[i].push(t);
        }
    }
    console.log("ROW DONE");
    let result = [];
    for(let i = 0; i < temp.length; i++){
        result.push([]);
        for(let j = 0; j < temp[i].length; j++){
            let t = {real:0,imag:0};
            for(let k = 0; k < temp.length; k++){
                let o = mulComp(doubleuNnk(temp.length,k,i),temp[k][j])
                t = addComp(t,o)
            }
            if(magComp(t) > max){
                max = magComp(t);
            }
            result[i].push(t);
        }
    }
    console.log("COLUMN DONE");
    return [result,max];
}
function draw(img) {
    var canvas = document.getElementById('canvas');
    canvas.height = img.height;
    canvas.width = img.width;
    var ctx = canvas.getContext('2d');
    console.log(img.height, img.width);
    ctx.drawImage(img, 0, 0);
    img.style.display = 'none';
    let dftResult;
```

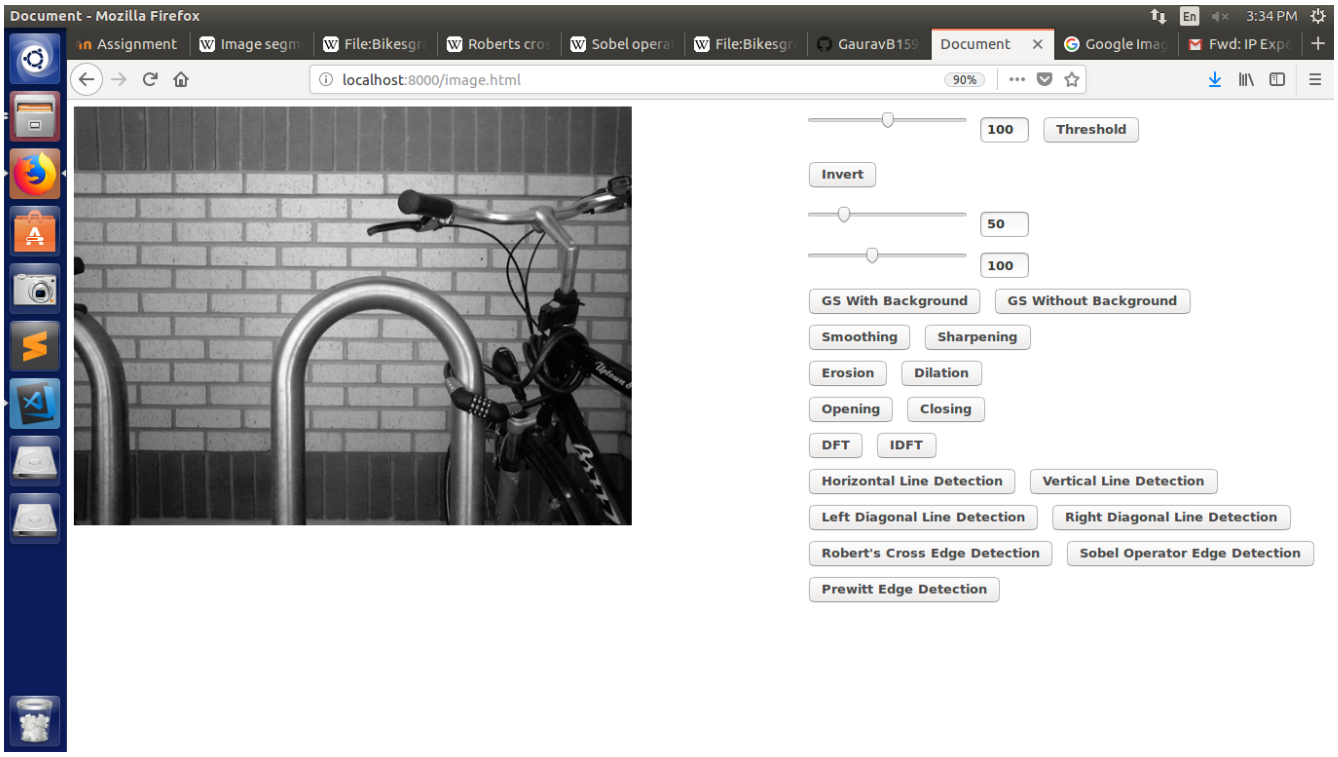
```
var imageData = ctx.getImageData(0, 0, canvas.width, canvas.height);
let height = img.height,
    width = img.width;
var data = imageData.data;
var invert = function() {
    for (var i = 0; i < data.length; i += 4) {
        data[i]     = 255 - data[i];     // red
        data[i + 1] = 255 - data[i + 1]; // green
        data[i + 2] = 255 - data[i + 2]; // blue
    }
    ctx.putImageData(imageData, 0, 0);
    data = imageData.data;
};
var threshold = function(){
    let t = parseInt(document.getElementById('threshold').value);
    for (var i = 0; i < data.length; i += 4) {
        if(data[i] < t){
            data[i] = 255;     // red
        }else{
            data[i] = 0;
        }
        if(data[i+1] < t){
            data[i+1] = 255;
        }else{
            data[i+1] = 0;
        }
        if(data[i+2] < t){
            data[i+2] = 255;
        }else{
            data[i+2] = 0;
        }
    }
    ctx.putImageData(imageData, 0, 0);
}
var glslicing = function(wb){
    let a = parseInt(document.getElementById('greya').value),
        b = parseInt(document.getElementById('greyb').value);
    if(a > b){
        let temp = a;
        a = b;
        b = temp;
    }
    for (var i = 0; i < data.length; i += 4) {
        if(data[i] < a || data[i] > b){
            data[i] = 255;
        }else if(wb){
            data[i] = 0;
        }
        if(data[i+1] < a || data[i+1] > b){
            data[i+1] = 255;
        }else if(wb){
            data[i+1] = 0;
        }
        if(data[i+2] < a || data[i+2] > b){
            data[i+2] = 255;
        }else if(wb){
            data[i+2] = 0;
        }
    }
```

```
            ctx.putImageData(imageData, 0, 0);
        }
        function twotoone(i,j,width){
            if(i < 0 || j < 0 || i >= width || j >= height){
                return -1;
            }
            return j*width + i;
        }
        var smoothingEv = function(){
            let dat = [],dat2=[];
            for(let i = 0; i < data.length; i+=4){
                dat.push(data[i]);
                dat2.push(data[i]);
            }
            let c;
            let mask = [[1, 1, 1], [1, 1, 1], [1, 1, 1]]
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask[k][l];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,
width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,
width)];

                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum /= 9;
                    sum = Math.round(sum);
                    dat2[twotoone(i+1,j+1,width)] = sum;
                }
            }
            // console.log(dat2);
            for(let i = 0; i < dat2.length;i++){
                data[4*i] = dat2[i];
                data[4*i+1] = dat2[i];
                data[4*i+2] = dat2[i];
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var sharpeningEv = function () {
            let dat = [], dat2 = [];
            for (let i = 0; i < data.length; i += 4) {
                dat.push(data[i]);
                dat2.push(data[i]);
            }
            let c;
            let mask = [[-1,-1,-1],
                        [-1,17,-1],
                        [-1,-1,-1]];
            // let mask = [[0, -1, 0],
            //             [-1, 5, -1],
            //             [0, -1, 0]]
            for (let i = -1; i < width; i++) {
                for (let j = -1; j < height; j++) {
                    let sum = 0;
                    for (let k = 0; k < 3; k++) {
                        for (let l = 0; l < 3; l++) {
                            c = mask[k][l];
                            if (dat[twotoone(i + k, j + l, width)]) {
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            } else {
                                if (j === -1 && l === 0) {
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,
width)];

                                } else if (i === -1 && k === 0) {
                                    sum += c*dat[twotoone(i + k + 1, j + l,
width)];

                                } else if (j === height && l === 2) {
                                    sum += c*dat[twotoone(i + k, j + l - 1,
width)];

                                }
                                else if (i === width && k === 2) {
                                    sum += c*dat[twotoone(i + k - 1, j + l,
width)];

                                }
                            }
                        }
                    }
                    sum /= 9;
                    sum = Math.round(sum);
                    if(sum < 0){
                        sum = 0;
                    }
                    dat2[twotoone(i + 1, j + 1, width)] = sum;
                }
            }
            // console.log(dat2);
            for (let i = 0; i < dat2.length; i++) {
                data[4 * i] = dat2[i];
                data[4 * i + 1] = dat2[i];
                data[4 * i + 2] = dat2[i];
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var dilationEv = function () {
            let dat = [], dat2 = [];
            for (let i = 0; i < data.length; i += 4) {
                dat.push(data[i]);
                dat2.push(data[i]);
            }
```

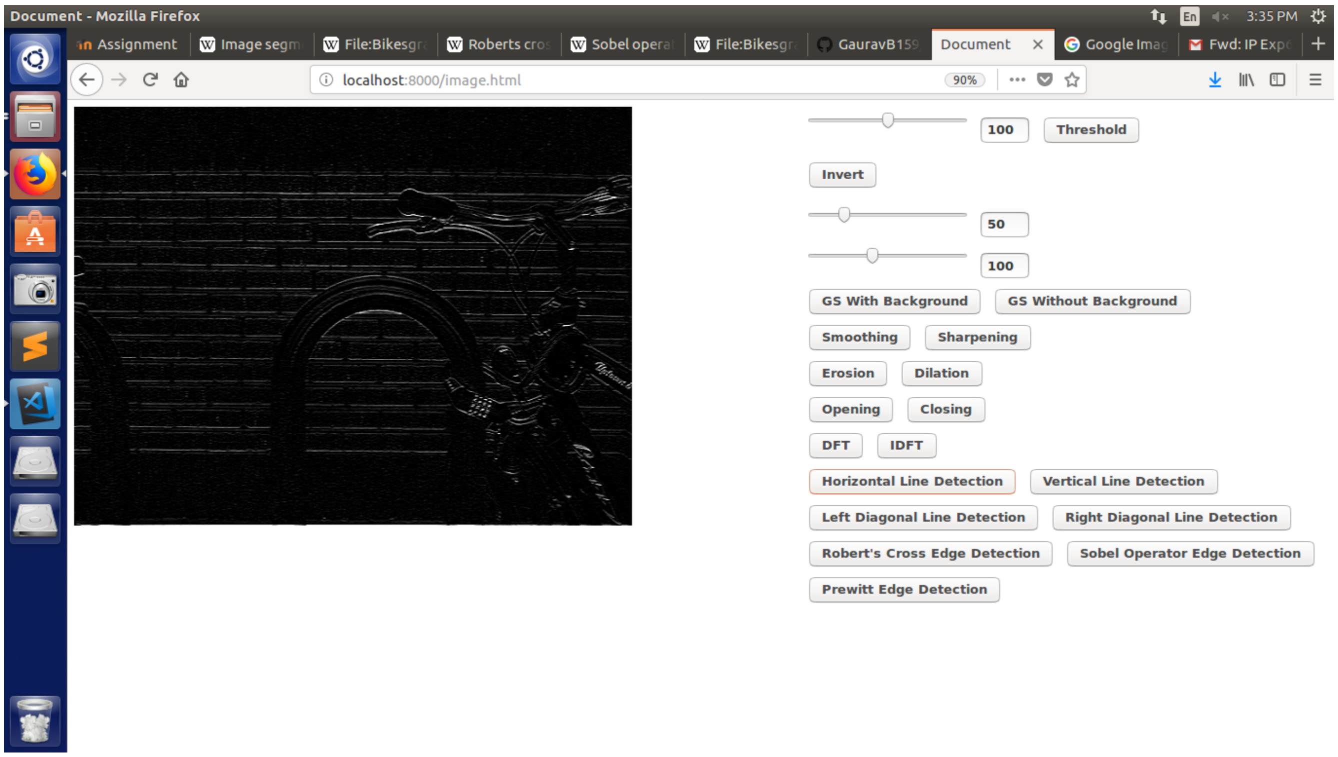
```
            let c;
            let mask = [
                [1, 1, 1, 1, 0],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [0, 1, 1, 1, 1]
            ];
            for (let i = 0; i < width; i++) {
                for (let j = 0; j < height; j++) {
                    let max = 0;
                    for (let k = -1; k < mask.length-1; k++) {
                        for (let l = -1; l < mask.length-1; l++) {
                            c = mask[k+1][l+1];
                            let hold = dat[twotoone(i + k, j + l, width)];
                            if (hold !== undefined && c === 1 && hold > max) {
                                max = hold;
                            }
                        }
                    }
                    dat2[twotoone(i, j, width)] = max;
                }
            }
            for (let i = 0; i < dat2.length; i++) {
                data[4 * i] = dat2[i];
                data[4 * i + 1] = dat2[i];
                data[4 * i + 2] = dat2[i];
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var erosionEv = function () {
            console.log("In")
            let dat = [], dat2 = [];
            for (let i = 0; i < data.length; i += 4) {
                dat.push(data[i]);
                dat2.push(data[i]);
            }
            let c;
            let mask = [
                [1, 1, 1, 1, 0],
                [1, 0, 1, 1, 1],
                [1, 1, 1, 1, 1],
                [1, 1, 1, 0, 1],
                [0, 1, 1, 1, 1]
            ];
            for (let i = 0; i < width; i++) {
                for (let j = 0; j < height; j++) {
                    let min = 255;
                    for (let k = -1; k < mask.length-1; k++) {
                        for (let l = -1; l < mask.length-1; l++) {
                            c = mask[k+1][l+1];
                            let hold = dat[twotoone(i + k, j + l, width)];
                            if (hold !== undefined && c === 1 && hold < min) {
                                min = hold;
                            }else if(hold === undefined){
                                // console.log("Hello",i,j,height,width,
dat.length, twotoone(i + k, j + l, width),min)
                            }
                        }
                    }
                    dat2[twotoone(i, j, width)] = min;
                }
            }
            for (let i = 0; i < dat2.length; i++) {
                data[4 * i] = dat2[i];
                data[4 * i + 1] = dat2[i];
                data[4 * i + 2] = dat2[i];
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var closingEv = function () {
            erosionEv();
            dilationEv();
        }
        var openingEv = function () {
            dilationEv();
            erosionEv();
        }
        var dft = function(){
            let dat = [[]], count = 0, count2 = 0;
            for (let i = 0; i < data.length; ) {
                if(count2 < width){
                    dat[count].push(data[i]);
                    count2++;
                    i += 4
                }else{
                    count++;
                    count2 = 0;
                    dat.push([]);
                }
            }
            let result = TwoDDFT(dat);
            let max = result[1];
            dftResult = result[0];
            let c = 255 / Math.log(1 + max);
            result = result[0];
            for (let i = 0; i < result.length; i++) {
                for(let j = 0; j < result[i].length;j++){
                    data[4 * (i*width+j)] = c*Math.log(1+magComp(result[i][j]));
                    data[4 * (i*width+j) + 1] = c*Math.log(1+magComp(result[i]
[j]));

                    data[4 * (i*width+j) + 2] = c*Math.log(1+magComp(result[i]
[j]));

                    // data[4 * (i*width+j)] = magComp(result[i][j]);
                    // data[4 * (i*width+j) + 1] = magComp(result[i][j]);
                    // data[4 * (i*width+j) + 2] = magComp(result[i][j]);
                }
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var idft = function(){
            if(dftResult){
                let result = TwoDIDFT(dftResult);
                for (let i = 0; i < result.length; i++) {
                    for(let j = 0; j < result[i].length;j++){
                        data[4 * (i*width+j)] = magComp(result[i][j]);
                        data[4 * (i*width+j) + 1] = magComp(result[i][j]);
                        data[4 * (i*width+j) + 2] = magComp(result[i][j]);
                    }
                }
                ctx.putImageData(imageData, 0, 0);
            }else{
                alert("DFT has not been applied");
            }

        }
        var convolve = function(mask){
            let dat = [],dat2=[];
            for(let i = 0; i < data.length; i+=4){
                dat.push(data[i]);
                dat2.push(data[i]);
            }
            let c;
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,
width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,

width)];
                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat2[twotoone(i+1,j+1,width)] = sum;
                }
            }
            // console.log(dat2);
            for(let i = 0; i < dat2.length;i++){
                data[4*i] = dat2[i];
                data[4*i+1] = dat2[i];
                data[4*i+2] = dat2[i];
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var horizontalLineDetection = function(){
            convolve([
                [-1, -1, -1],
                [2, 2, 2],
                [-1, -1, -1]
            ])
        }
        var verticalLineDetection = function(){
            convolve([
                [-1, 2, -1],
                [-1, 2, -1],
                [-1, 2, -1]
            ])
        }
        var leftDiagonalLineDetection = function(){
            convolve([
                [2, -1, -1],
                [-1, 2, -1],
                [-1, -1, 2]
            ])
        }
        var rightDiagonalLineDetection = function(){
            convolve([
                [-1, -1, 2],
                [-1, 2, -1],
                [2, -1, -1]
            ])
        }
        var sobel = function(){
            let dat = [],dat2=[], dat3=[];
            for(let i = 0; i < data.length; i+=4){
                dat.push(data[i]);
                dat2.push(data[i]);
                dat3.push(data[i]);
            }
            let c;
            let mask = [
                [1, 2, 1],
                [0, 0, 0],
                [-1, -2, -1]
            ];
            let mask2 = [
                [1, 0, -1],
                [2, 0, -2],
                [1, 0, -1]
            ];
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,
width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,
width)];

                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat2[twotoone(i+1,j+1,width)] = sum;
                }
            }
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask2[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,

width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,

width)];

                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat3[twotoone(i+1,j+1,width)] = sum;
                }
            }
            // console.log(dat2);
            for(let i = 0; i < dat2.length;i++){
                let temp = Math.sqrt(dat2[i]*dat2[i]+dat3[i]*dat3[i]);
                // let temp = dat2[i]+dat3[i];
                data[4*i] = temp;
                data[4*i+1] = temp;
                data[4*i+2] = temp;
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var prewitt = function(){
            let dat = [],dat2=[], dat3=[];
            for(let i = 0; i < data.length; i+=4){
                dat.push(data[i]);
                dat2.push(data[i]);
                dat3.push(data[i]);
            }
            let c;
            let mask = [
                [1, 1, 1],
                [0, 0, 0],
                [-1, -1, -1]
            ];
            let mask2 = [
                [1, 0, -1],
                [1, 0, -1],
                [1, 0, -1]
            ];
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,

width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,

width)];
                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat2[twotoone(i+1,j+1,width)] = sum;
                }
            }
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < 3; k++){
                        for(let l = 0; l < 3; l++){
                            c = mask2[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,

width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,

width)];
                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat3[twotoone(i+1,j+1,width)] = sum;
                }
            }
            // console.log(dat2);
            for(let i = 0; i < dat2.length;i++){
                let temp = Math.sqrt(dat2[i]*dat2[i]+dat3[i]*dat3[i]);
                // let temp = dat2[i]+dat3[i];
                data[4*i] = temp;
                data[4*i+1] = temp;
                data[4*i+2] = temp;
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var roberts = function(){
            let dat = [],dat2=[], dat3=[];
            for(let i = 0; i < data.length; i+=4){
                dat.push(data[i]);
                dat2.push(data[i]);
                dat3.push(data[i]);
            }
            let c;
            let mask = [
                [1, 0],
                [0, -1]
            ];
            let mask2 = [
                [0, 1],
                [-1, 0]
            ];
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < mask.length; k++){
                        for(let l = 0; l < mask.length; l++){
                            c = mask[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,

width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,

width)];

                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat2[twotoone(i+1,j+1,width)] = sum;
                }
            }
            for(let i = -1; i < width;i++){
                for(let j = -1; j < height; j++){
                    let sum = 0;
                    for(let k = 0; k < mask.length; k++){
                        for(let l = 0; l < mask.length; l++){
                            c = mask2[l][k];
                            if(dat[twotoone(i + k, j + l, width)]){
                                sum += c*dat[twotoone(i + k, j + l, width)];
                            }else{
                                if(j === -1 && l === 0){
                                    // console.log(twotoone(i + k, j + l + 1,
width),j,l,i)

                                    sum += c*dat[twotoone(i + k, j + l + 1,
width)];

                                }else if(i === -1 && k === 0){
                                    sum += c*dat[twotoone(i + k + 1, j + l,
width)];

                                }else if(j === height && l === 2){
                                    sum += c*dat[twotoone(i + k, j+l-1 , width)];
                                }
                                else if(i === width && k === 2){
                                    sum += c*dat[twotoone(i+k-1, j + l, width)];
                                }
                            }
                        }
                    }
                    sum = Math.round(sum);
                    dat3[twotoone(i+1,j+1,width)] = sum;
                }
            }
            // console.log(dat2);
            for(let i = 0; i < dat2.length;i++){
                let temp = Math.sqrt(dat2[i]*dat2[i]+dat3[i]*dat3[i]);
                // let temp = dat2[i]+dat3[i];
                data[4*i] = temp;
                data[4*i+1] = temp;
                data[4*i+2] = temp;
            }
            ctx.putImageData(imageData, 0, 0);
        }
        var invertbtn = document.getElementById('invertbtn');
        invertbtn.addEventListener('click', invert);
        var thresholdbtn = document.getElementById('thresholdapply');
        thresholdbtn.addEventListener('click', threshold);
        var gsb = document.getElementById('gsb');
        gsb.addEventListener('click', ()=>{ glslicing(false) });
        var gswb = document.getElementById('gswb');
        gswb.addEventListener('click', ()=>{ glslicing(true) });
        var smoothing = document.getElementById('smoothing');
        smoothing.addEventListener('click', ()=>{ smoothingEv() });
        var sharpening = document.getElementById('sharpening');
        sharpening.addEventListener('click', () => { sharpeningEv() });
        var dilation = document.getElementById('dilation');
        dilation.addEventListener('click', () => { dilationEv() });
        var erosion = document.getElementById('erosion');
        erosion.addEventListener('click', () => { erosionEv() });
        var closing = document.getElementById('closing');
        closing.addEventListener('click', () => { closingEv() });
        var opening = document.getElementById('opening');
        opening.addEventListener('click', () => { openingEv() });
        var fourier = document.getElementById('fourier');
        fourier.addEventListener('click',() => { dft() })
        var invfourier = document.getElementById('invfourier');
        invfourier.addEventListener('click',() => { idft() })
        var horizLine = document.getElementById('horiz');
        horizLine.addEventListener('click',() => { horizontalLineDetection() })
        var vertLine = document.getElementById('vert');
        vertLine.addEventListener('click',() => { verticalLineDetection() })
        var ldiagLine = document.getElementById('ldiag');
        ldiagLine.addEventListener('click',() => { leftDiagonalLineDetection() })
        var rdiagLine = document.getElementById('rdiag');
        rdiagLine.addEventListener('click',() => { rightDiagonalLineDetection() })
        var sobelEdge = document.getElementById('sobel');
        sobelEdge.addEventListener('click',() => { sobel() })
        var robertEdge = document.getElementById('robert');
        robertEdge.addEventListener('click',() => { roberts() })
        var prewittEdge = document.getElementById('prewitt');
        prewittEdge.addEventListener('click',() => { prewitt() })
    }

    </script>
</body>
</html>
```

Output:

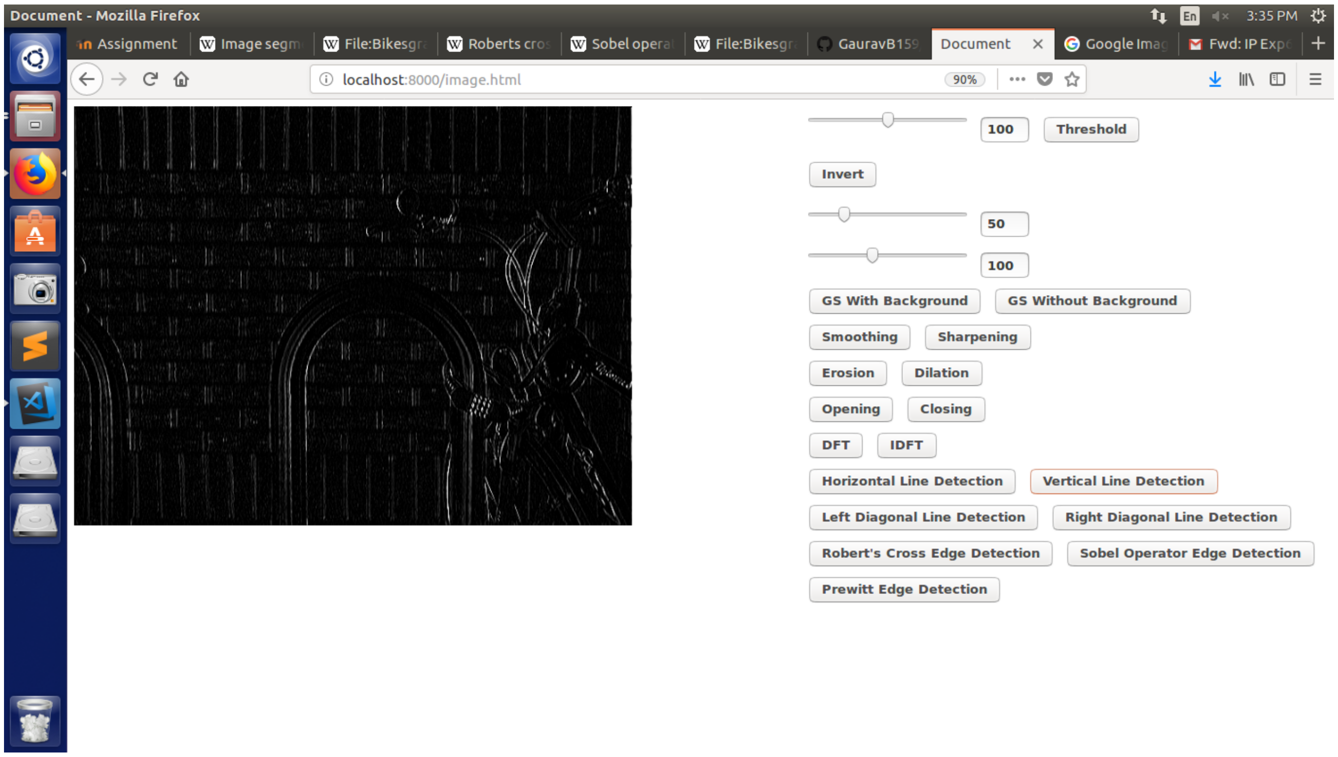
Original Image:



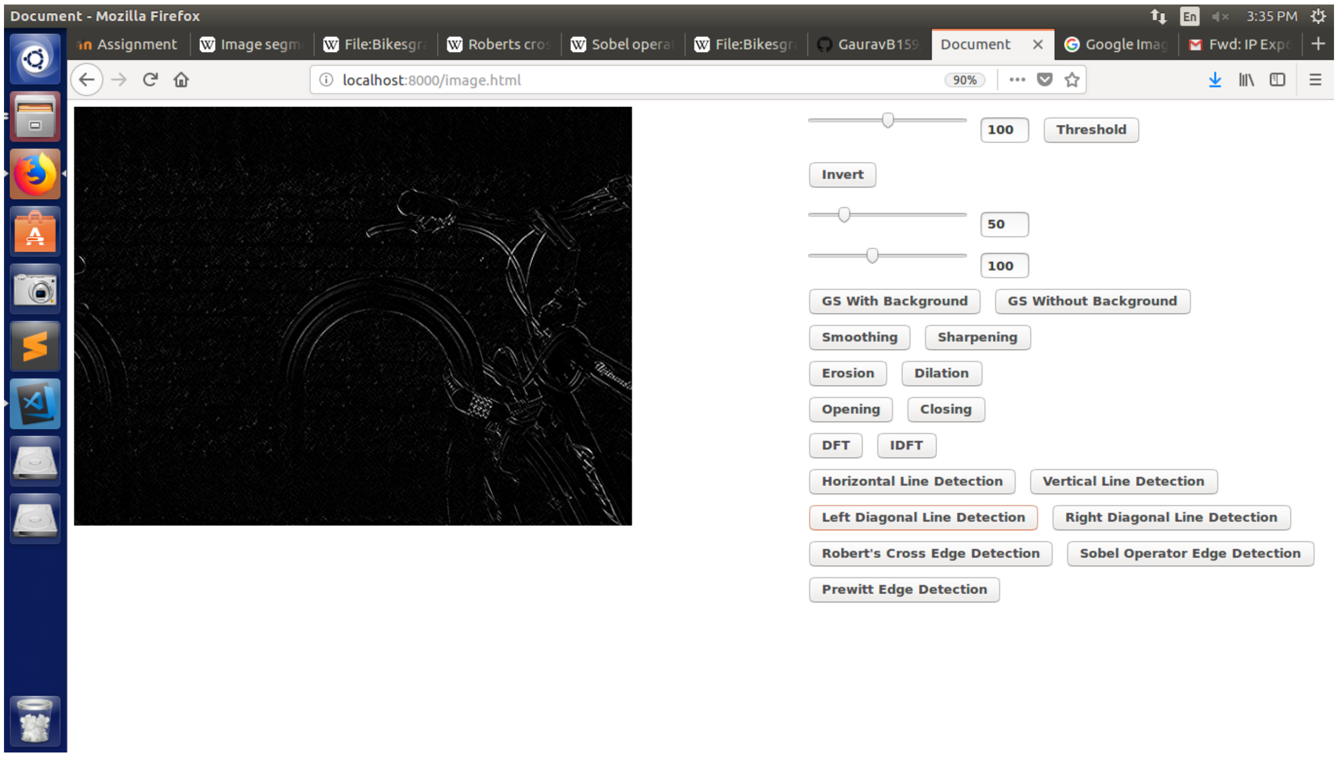
Horizontal Line Detection:



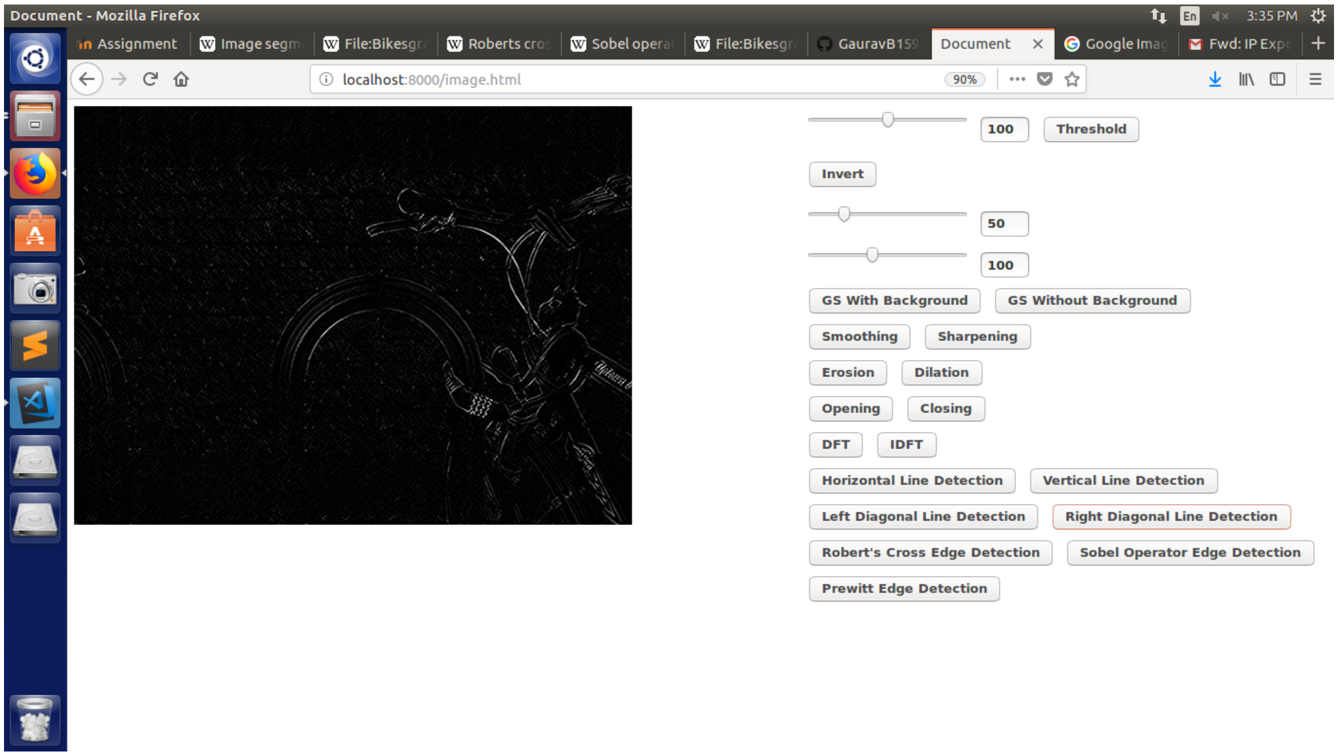
Vertical Line Detection:


Document - Mozilla Firefox
localhost:8000/image.html
100
Threshold
Invert
50
100
GS With Background
GS Without Background
Smoothing
Sharpening
Erosion
Dilation
Opening
Closing
DFT
IDFT
Horizontal Line Detection
Vertical Line Detection
Left Diagonal Line Detection
Right Diagonal Line Detection
Robert's Cross Edge Detection
Sobel Operator Edge Detection
Prewitt Edge Detection

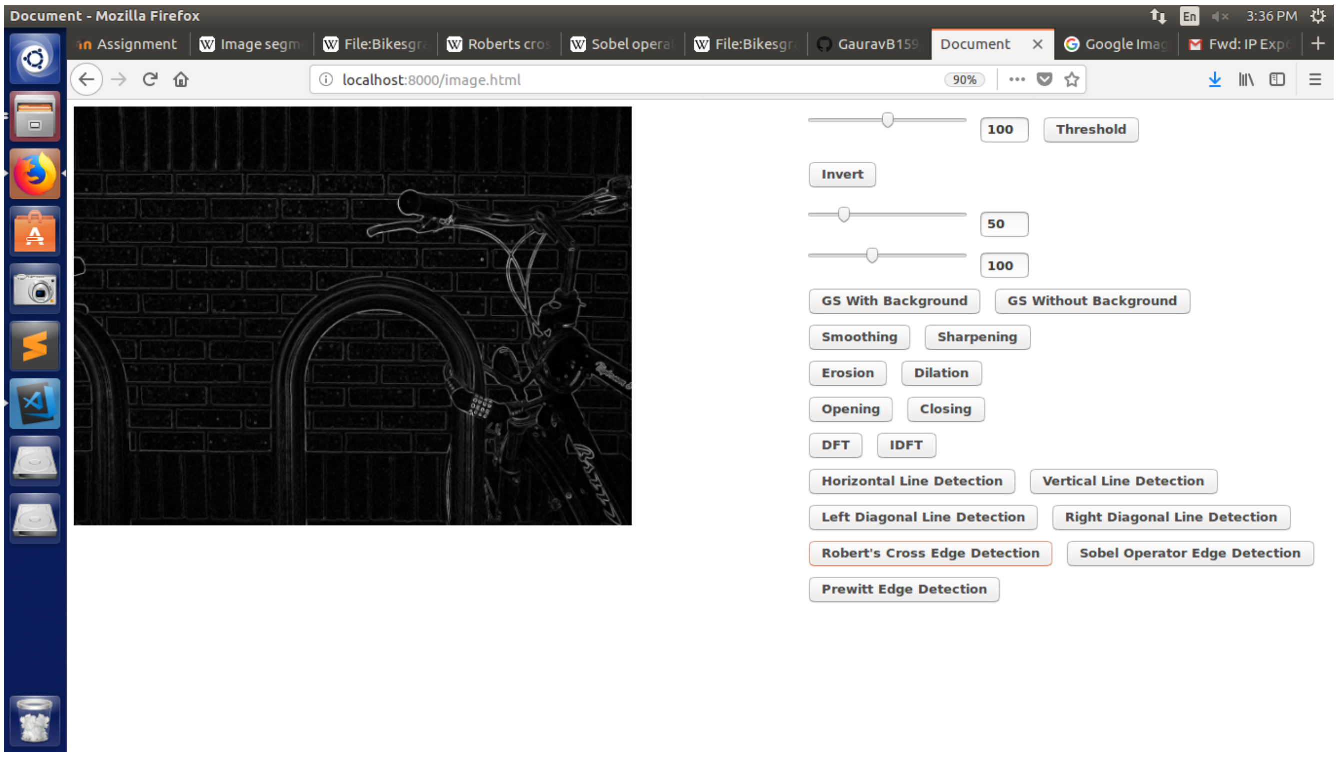
Left Diagonal Line Detection:


Document - Mozilla Firefox
localhost:8000/image.html
100
Threshold
Invert
50
100
GS With Background
GS Without Background
Smoothing
Sharpening
Erosion
Dilation
Opening
Closing
DFT
IDFT
Horizontal Line Detection
Vertical Line Detection
Left Diagonal Line Detection
Right Diagonal Line Detection
Robert's Cross Edge Detection
Sobel Operator Edge Detection
Prewitt Edge Detection

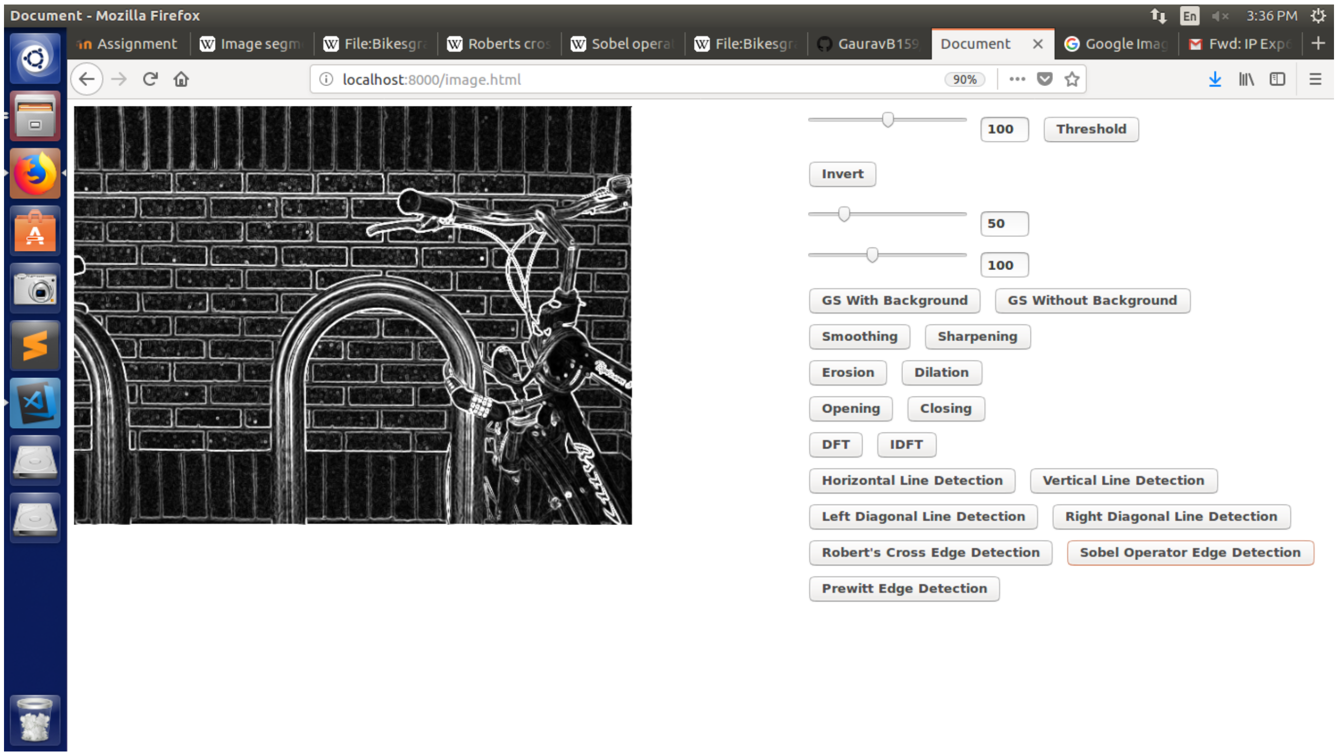
Right Diagonal Line Detection:


Document - Mozilla Firefox
localhost:8000/image.html
100
Threshold
Invert
50
100
GS With Background
GS Without Background
Smoothing
Sharpening
Erosion
Dilation
Opening
Closing
DFT
IDFT
Horizontal Line Detection
Vertical Line Detection
Left Diagonal Line Detection
Right Diagonal Line Detection
Robert's Cross Edge Detection
Sobel Operator Edge Detection
Prewitt Edge Detection

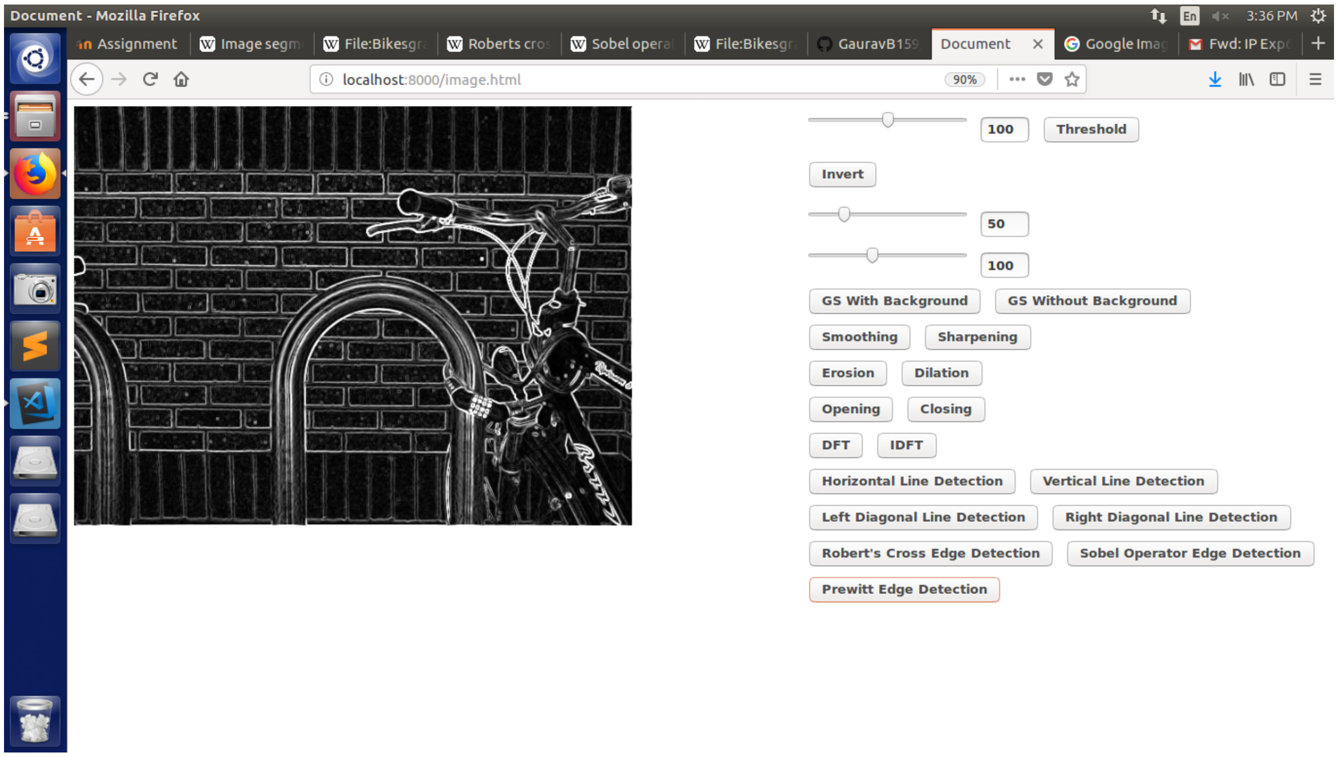
Robert’s Cross Edge Detection:


Document - Mozilla Firefox
localhost:8000/image.html
100
Threshold
Invert
50
100
GS With Background
GS Without Background
Smoothing
Sharpening
Erosion
Dilation
Opening
Closing
DFT
IDFT
Horizontal Line Detection
Vertical Line Detection
Left Diagonal Line Detection
Right Diagonal Line Detection
Robert's Cross Edge Detection
Sobel Operator Edge Detection
Prewitt Edge Detection

Sobel Operator Edge Detection:



Prewitt Edge Detection:



Conclusion: Applied various edge and line detection techniques