PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

Caracterización y clasificación automática de ríos en imágenes satelitales

ANEXOS

Tesis para optar el Título de Ingeniero Informático, que presenta el bachiller:

Kevin Brown Manrique

Asesor: Dr. César Armando Beltrán Castañón

Lima, Junio del 2017

Índice

1 Código Desarrollado

1.1 Función clean

```
function bw = clean(bwImage)

        n = 5; %definir número de veces que se efectuará la operación de spur (eliminar pixeles terminales)

        bw = bwImage;

        for i=1:n

                bw = bwmorph(bw,'spur');

        end

end
```

1.2 Función readImages

```
function imgs = readImages(routeName) % routeName = ruta que contiene las imágenes de los esqueletos

        files = dir(strcat(routeName,'*.png')); %Definir que se leerán sólo imágenes con terminación .png

        %generar matriz para guardar data

        % fila1 = matriz de imagen

        % fila2 = matriz bw de imagen

        % fila3 = area

        % fila4 = perimetro

        % fila5 = matriz de bifurcaciones

        % fila6 = cantidad de bifurcaciones

    c = cell(6,length(files));

    for k = 1:length(files) %Realiza las siguientes operaciones según el número de archivos guardados en el directorio

        fileName = files(k).name; %Obtiene el nombre de los archivos

        I = imread(strcat(routeName,fileName)); %Lee la imagen de orden k

        BW = im2bw(I); % Convierte la imagen a binaria

        c{1,k} = I; % Almacena la imagen (pixeles) en la fila1

        c{2,k} = BW; % Almacena la imagen binarizada (pixeles) en la fila2

        perim = regionprops(BW,'perimeter'); % Calcula el perimetro

        area = regionprops(BW,'area'); % Calcula el área

        c{3,k} = area.Area; % Almacena el área en la fila3
```

```
        c{4,k} = perim.Perimeter; % Almacena el perímetro en la fila4
        aux = bifurcationMatrix(BW); % Utiliza la función bifurcationMatrix
        c{5,k} = aux; % Almacena la matriz de bifurcaciones en la fila 5
        c{6,k} = numel(aux(aux==3)) + numel(aux(aux==4)); % Almacena la suma del
número de bifurcaciones y cruces encontrados
    end
    imgs = c; % retorna el arreglo generado
end
```

1.3 Función bifurcationMatrix

```
function bifMatrix = bifurcationMatrix(bwImage)
%CN = 0.5 * sumatoria(i=1 hasta 8 de: abs(P_i - P_i+1) donde P_i es el
%pixel en la vencidad de P con P_i = (0 o 1) y P_9 = P_1
%considerando:
%    P_4   P_3   P_2
%    P_5   P     P_1
%    P_6   P_7   P_8
test = bwImage;
[dimX dimY] = size(bwImage); % Se calculan dimensiones de la imagen de entrada
testValues=zeros([dimX dimY]); % Se genera una matriz de ceros con las
dimensiones halladas en la línea anterior
     for i = 1:dimX
      for j=1:dimY
         sum = 0;
        if (test(i,j) == 1) % Para cada pixel activo de la imagen de entrada se realiza
el método de Crossing Number
            if ((i==1) && (j==1))
               sum = 0.5 * (abs(test(i+1,j)-test(i+1,j+1))+ abs(test(i+1,j+1)-test(i,j+1)));
            elseif ((i==1)&&(j==dimY))
               sum = 0.5 * (abs(test(i,j-1)-test(i+1,j-1))+ abs(test(i+1,j-1)-test(i+1,j)));
            elseif ((i == dimX)&&(j==dimY));
               sum = 0.5 * (abs(test(i-1,j)-test(i-1,j-1))+ abs(test(i-1,j-1)-test(i,j-1)));
            elseif ((i == dimX) && (j==1))
```

```
            sum = 0.5 * (abs(test(i,j+1)-test(i-1,j+1))+ abs(test(i-1,j+1)-test(i-1,j)));
        elseif ((i ==1 ) && ((j>1)&&(j<dimY)))
            sum = 0.5 * (abs(test(i,j-1)-test(i+1,j-1))+ abs(test(i+1,j-1)-test(i+1,j))+
abs(test(i+1,j)-test(i+1,j+1))+ abs(test(i+1,j+1)-test(i,j+1)));
        elseif (((i>1)&&(i<dimX))&&(j==dimY))
            sum = 0.5 * (abs(test(i-1,j)-test(i-1,j-1))+ abs(test(i-1,j-1)-test(i,j-1))+
abs(test(i,j-1)-test(i+1,j-1))+ abs(test(i+1,j-1)-test(i+1,j)));
        elseif ((i == dimX ) && ((j>1)&&(j<dimY)))
            sum = 0.5 * (abs(test(i,j+1)-test(i-1,j+1))+ abs(test(i-1,j+1)-test(i-1,j))+
abs(test(i-1,j)-test(i-1,j-1))+ abs(test(i-1,j-1)-test(i,j-1)));
        elseif (((i>1)&&(i<dimX))&&(j==1))
            sum = 0.5 * (abs(test(i+1,j)-test(i+0,j+1))+ abs(test(i+1,j+1)-test(i,j+1))+
abs(test(i,j+1)-test(i-1,j+1))+ abs(test(i-1,j+1)-test(i-1,j))) ;
        else
            sum = 0.5 * (abs(test(i,j+1)-test(i-1,j+1))+ abs(test(i-1,j+1)-test(i-1,j))+
abs(test(i-1,j)-test(i-1,j-1))+ abs(test(i-1,j-1)-test(i,j-1))+ abs(test(i,j-1)-test(i+1,j-1))+
abs(test(i+1,j-1)-test(i+1,j))+ abs(test(i+1,j)-test(i+1,j+1))+ abs(test(i+1,j+1)-
test(i,j+1)));
        end
      testValues(i,j) = sum;
      end
    end
   end
 bifMatrix = testValues;
end
```

1.4 Función deriveAndMore

function featurePoints = deriveAndMore(rivers,distances) % Recibe como entrada el arreglo que contiene los ríos de una clase (obtenido de la función readImages) y un arreglo con las distancias a dilatar

```
  featurePoints = []; % Se crea un vector vacío
  colorMap = rand(size(rivers,2),3); % Se crea una matriz de dimensiones n x m;
```

donde n es el número de ríos en el arreglo recibido y m es = 3 para generar colores aleatorios que grafiquen a cada río

```
  hold on;
  for i=1:size(rivers,2) % A cada río se le aplicarán las siguientes operaciones
```

```
X = [] ; Y = [];
bw = rivers{2,i};
properties = cell(3,size(distances,2));
for k = 1:size(distances,2) %Se realizarán las dilataciones consecutivas según el vector distances recibido
    bwAux = bwdist(bw) <= distances(1,k); % dilata una distancia D_i / D_i = distances(1,i)
    perim = regionprops(bwAux,'perimeter'); % Se calcula el perimetro
    area = regionprops(bwAux,'area'); % Se calcula el área
    properties{1,k} = bwAux;
    properties{2,k} = area.Area;
    properties{3,k} = perim.Perimeter;
end
for z=1:size(distances,2)
    X = [X log(distances(1,z))]; %log(d)
    Y = [Y log(properties{2,z})]; %log(A)
end
S = spline(X,Y); % Con los puntos contenidos en X e Y se realiza una aproximación a una función
M = diag(3:-1:1,1); % Para efectuar una derivación se multiplican los coeficientes de la función por esta matriz

%Primera derivada
S1 = S;
S1.coefs = S1.coefs*M;

%Valores de la derivada
YY2 = [];
for n=1:size(X,2) %Se calculan los valores en la función derivada según los X
    YY2 = [YY2 fnval(S1,X(1,n))];
end

Xnorm = (X-min(X))/(max(X)-min(X)); % Se realiza una normalización a X
```

plot(Xnorm,YY2,1,colorMap(i)); % Se grafica la Curva Fractal: ejeX = normalizada de log(d); ejeY = dlog(d)/dlog(A)

%Puntos de interes

YInterest = [];

XInterest = [];

YY2aux = YY2;

X2norm = Xnorm;

YY2aux(size(YY2aux,2)) = [];

X2norm(size(X2norm,2)) = [];

YY2aux(1) = [];

X2norm(1) = [];

[xAux,yAux] = find(YY2aux == min(YY2aux)); %Se calcula el primer punto de interés: la máxima depresión

YInterest = [YInterest YY2aux(xAux,yAux)];

XInterest = [XInterest X2norm(xAux,yAux)];

YY2aux(yAux) = [];

X2norm(yAux) = [];

[xAux,yAux] = find(YY2aux == max(YY2aux)); %Se calcula el segundo punto de interés: el máximo pico

YInterest = [YInterest YY2aux(xAux,yAux)];

XInterest = [XInterest X2norm(xAux,yAux)];

YY2aux(yAux) = [];

X2norm(yAux) = [];

[xAux,yAux] = find(YY2aux == max(YY2aux)); %Se calcula el tercer punto de interés: el segundo máximo pico

YInterest = [YInterest YY2aux(xAux,yAux)];

XInterest = [XInterest X2norm(xAux,yAux)];

YY2aux(yAux) = [];

```
        X2norm(yAux) = [];
        plot(XInterest,YInterest,'*');

        featureVector = [XInterest' ; YInterest']; % transpuesta de X e Y para crear columna con caracteristicas
        featurePoints = [featurePoints featureVector];

    end
    hold off;
end
```

1.5 Módulo Principal

```
meandros = readImages('RUTA'); %lee imagenes contenidas en la ruta, convierte a binaria, halla area, perimetro y cant de bifurcaciones
m1 = zeros(1,size(meandros,2)); %clase 0
bifAux = cell2mat(meandros(6,:)); %obtiene en número de bifurcaciones para cada río
m1 = [m1 ; bifAux];

rectos = readImages('RUTA');
m2 = zeros(1,size(rectos,2));
m2 = m2+1; % clase 1
bifAux = cell2mat(rectos(6,:));
m2 = [m2 ; bifAux];

multicanal = readImages('RUTA');
m3 = zeros(1,size(multicanal,2));
m3 = m3+2; % clase 2
bifAux = cell2mat(multicanal(6,:));
m3 = [m3 ; bifAux];

distances = [1 sqrt(2) sqrt(3) 2*sqrt(2) 2 3 4 5]; % genera el vector de distancias a dilatar
distances = sort(distances); %ordena de manera creciente
```

subplot(1,4,1); %Separa la cuadricula donde se graficara en 1 fila con 4 columnas

featureMeandros = deriveAndMore(meandros,distances); %dilata isotropicamente y grafica curvas log(d)/log(A(d)) y la derivada de esta

m1 = [m1 ; featureMeandros]; %vector de características clase 0: meandros

subplot(1,4,2);

featureRectos = deriveAndMore(rectos,distances);

m2 = [m2 ; featureRectos]; %vector de características clase 1: rectos

subplot(1,4,3);

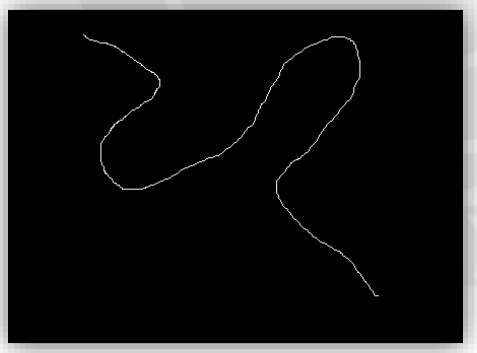
featureMulticanal = deriveAndMore(multicanal,distances);

m3 = [m3 ; featureMulticanal]; %vector de características clase 2: multicanal

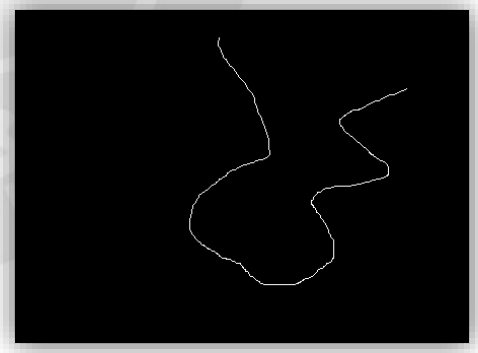
featureMatrix = [m1 m2 m3]; %matriz de características que entrará al clasificador

1.6 Esqueletos ríos meandros

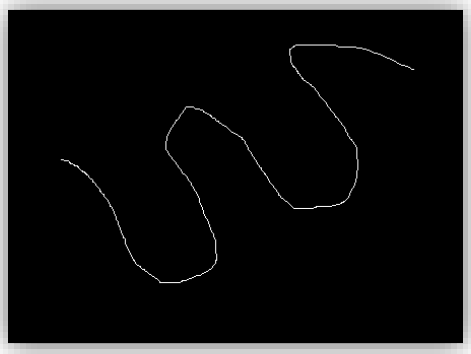
Río meandro 1



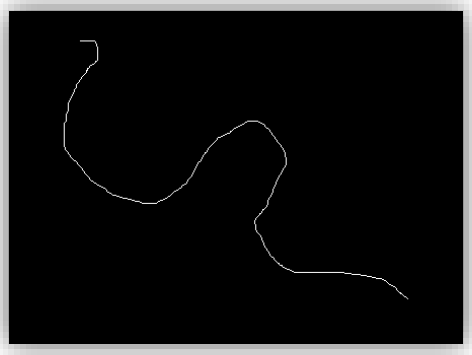
Río meandro 2

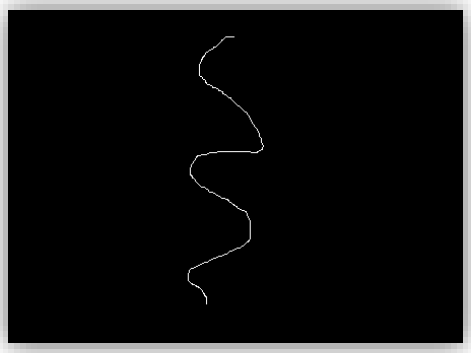
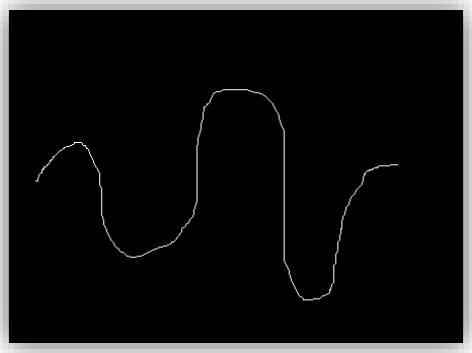
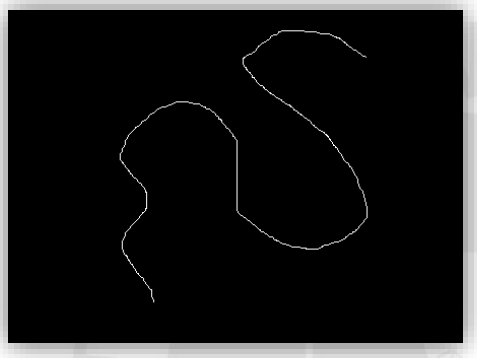
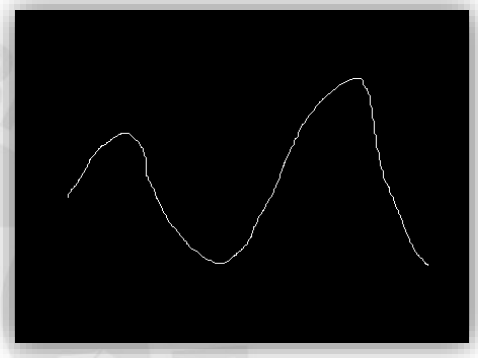
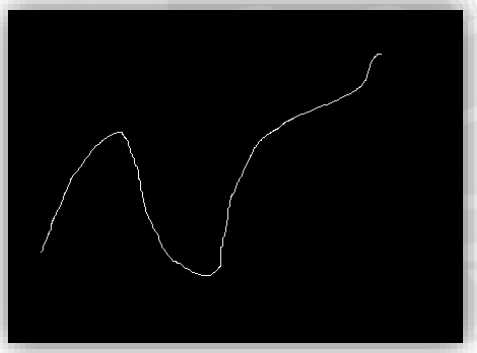
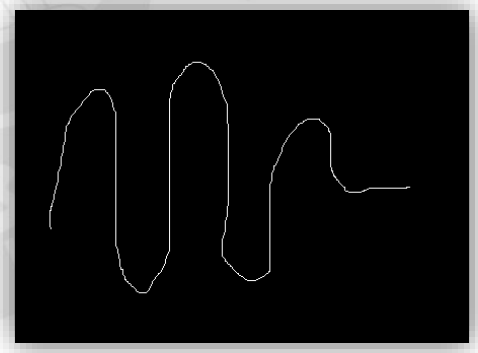
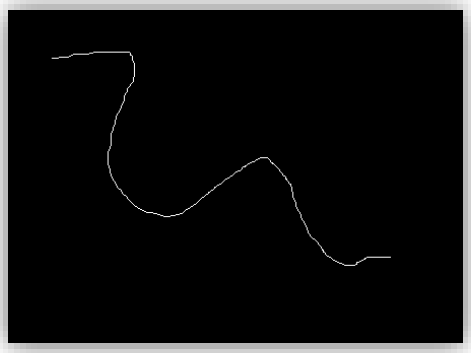
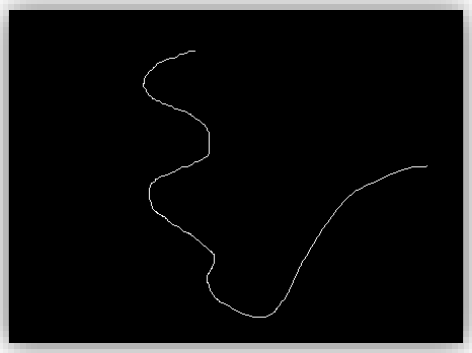


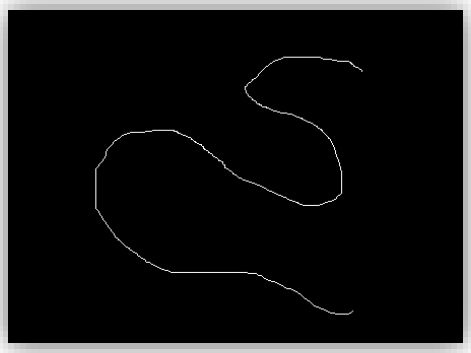
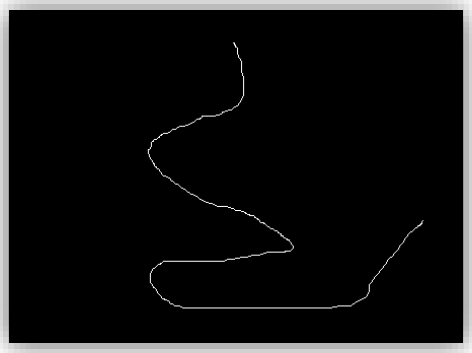
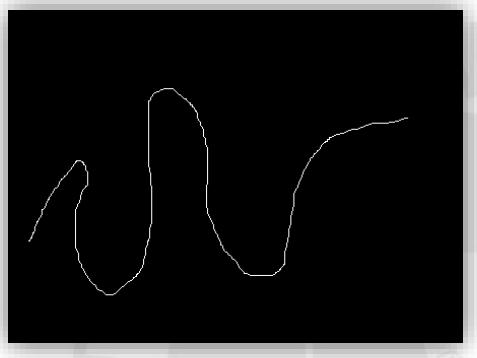
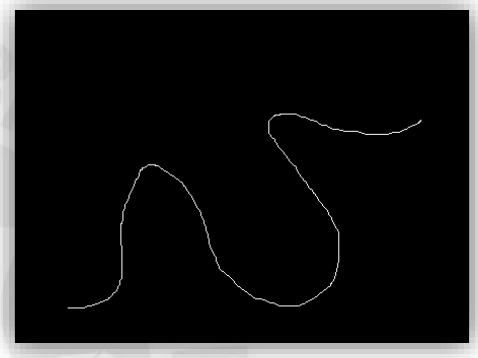
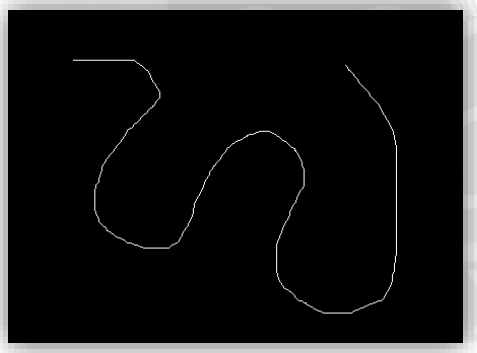
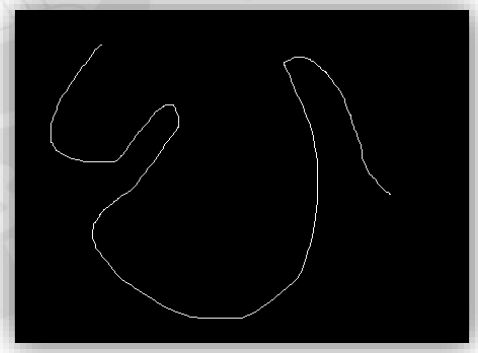
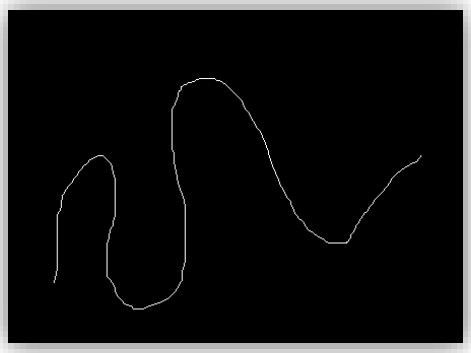
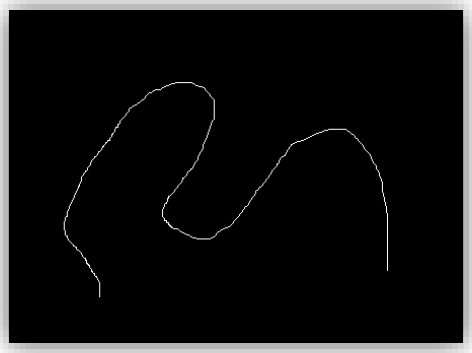
Río meandro 3

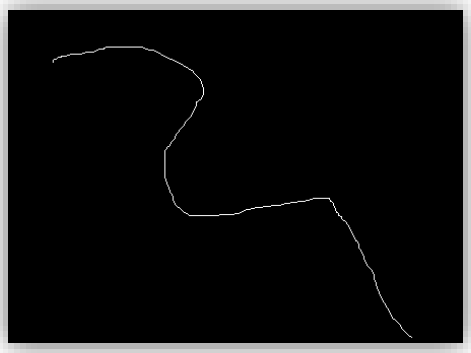
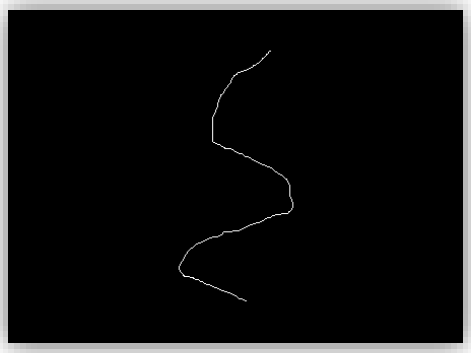
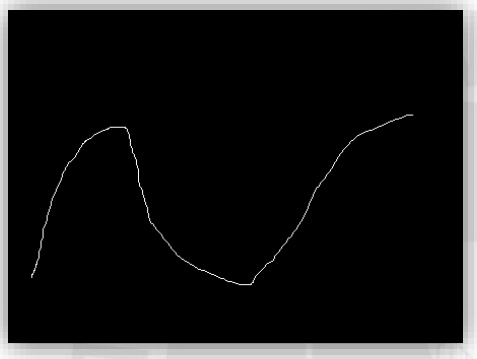
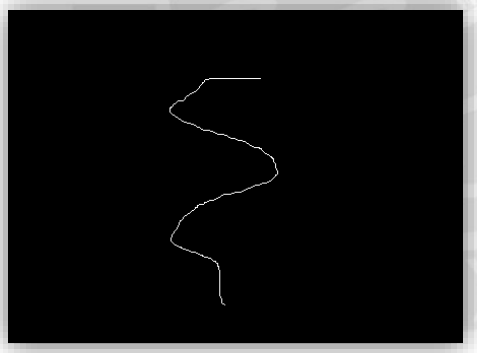
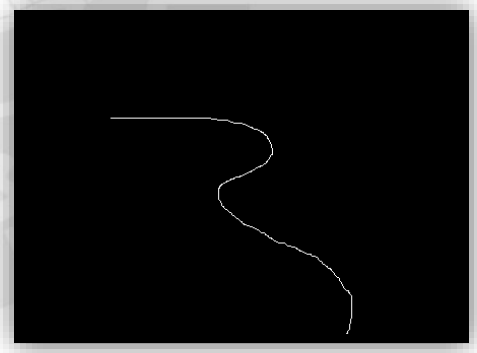
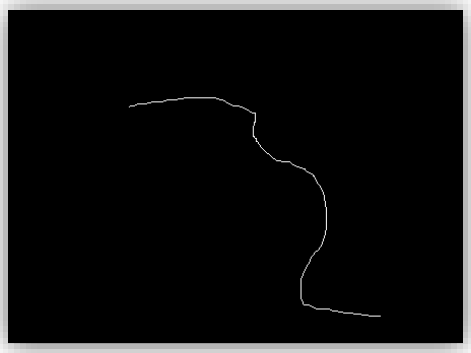
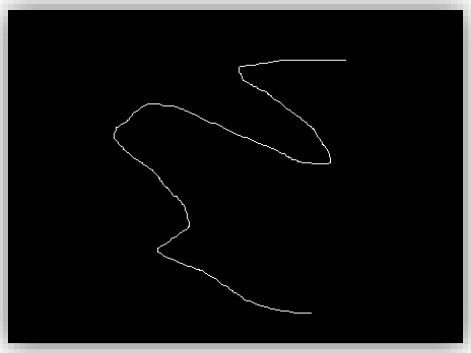


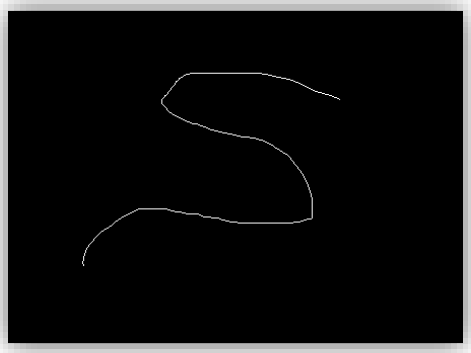
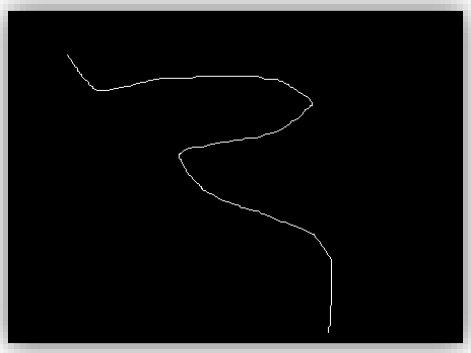
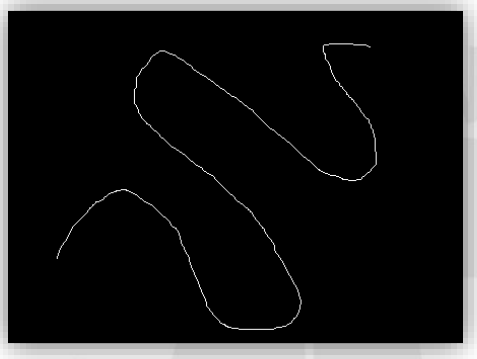
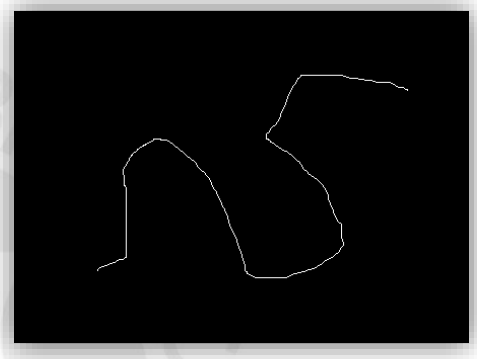
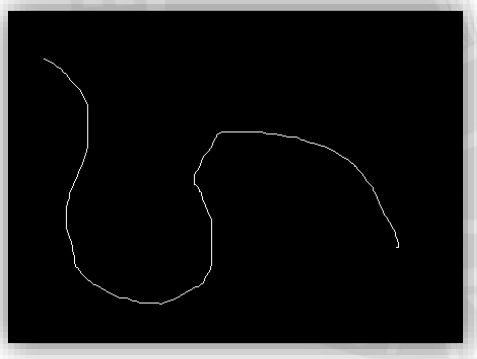
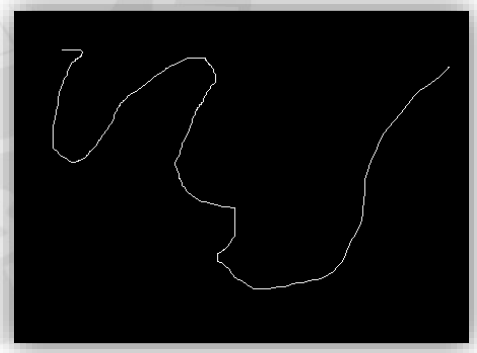
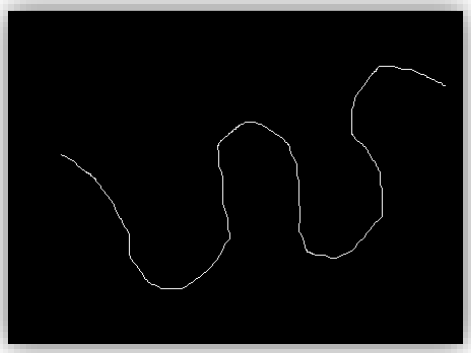
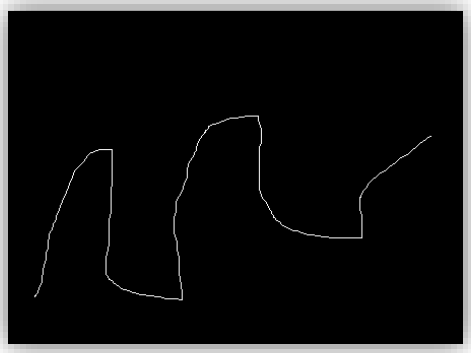
Río meandro 4

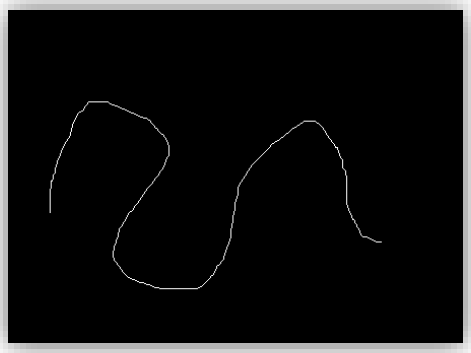
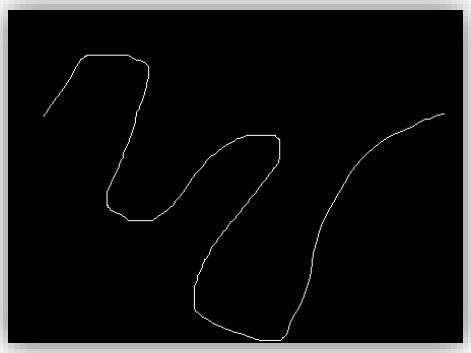
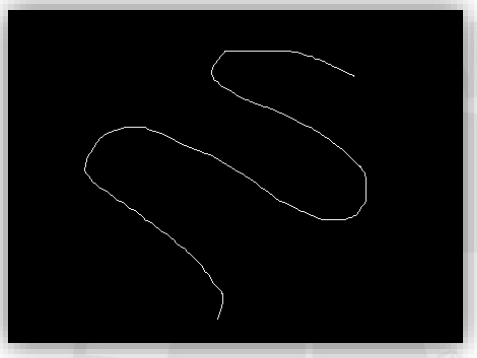
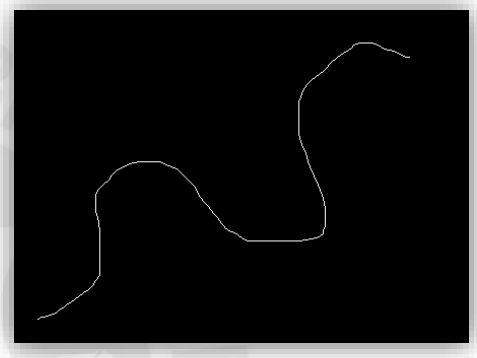


Río meandro 5 	Río meandro 6 
Río meandro 7 	Río meandro 8 
Río meandro 9 	Río meandro 10 
Río meandro 11 	Río meandro 12 

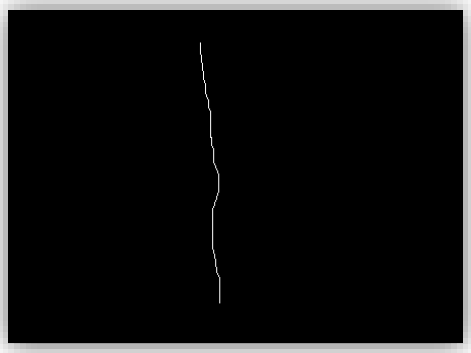
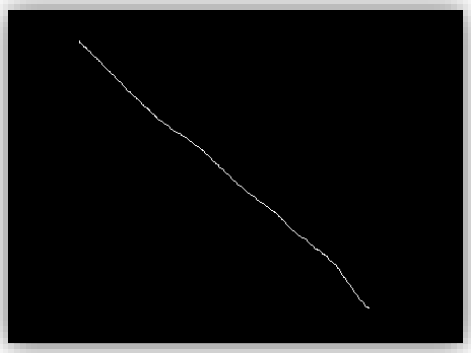
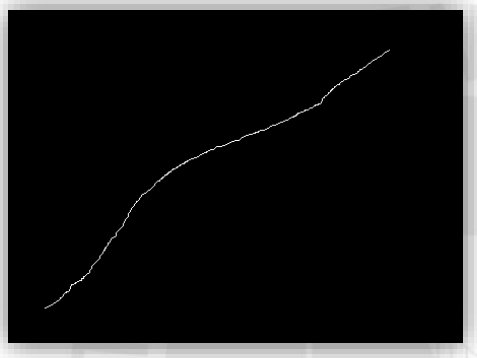
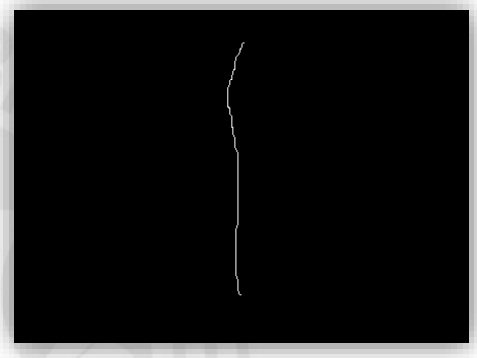
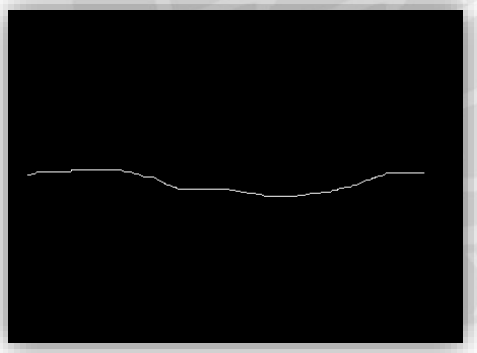
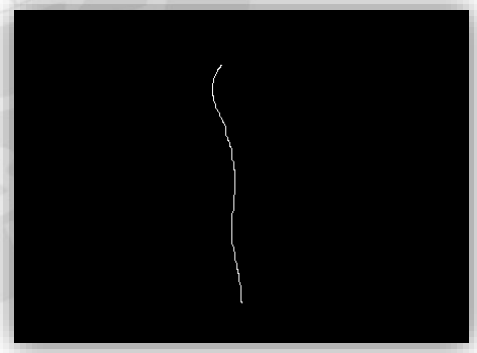
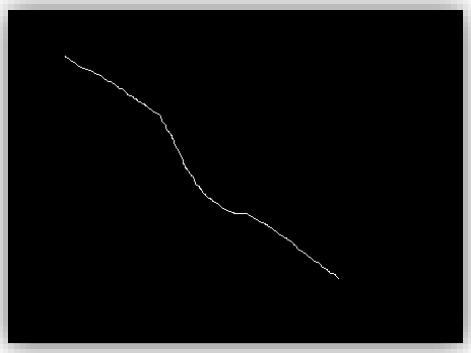
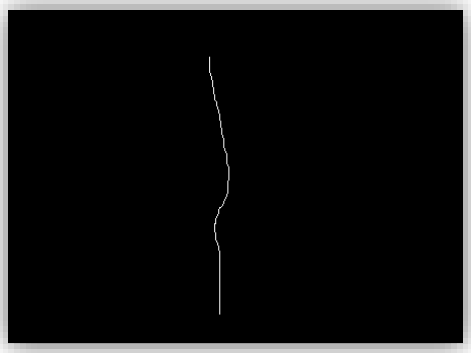
Río meandro 13 	Río meandro 14 
Río meandro 15 	Río meandro 16 
Río meandro 17 	Río meandro 18 
Río meandro 19 	Río meandro 20 

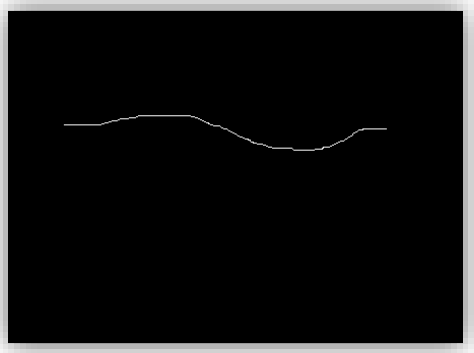
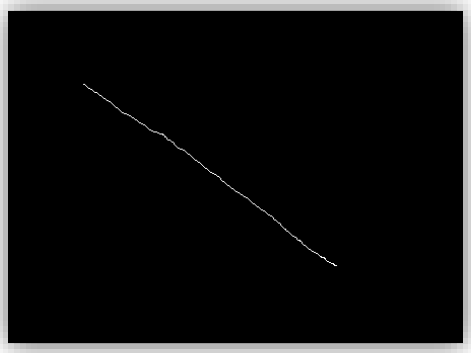
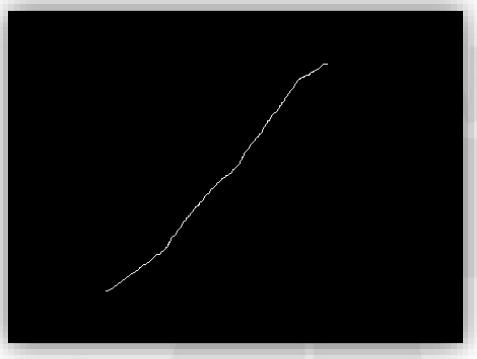
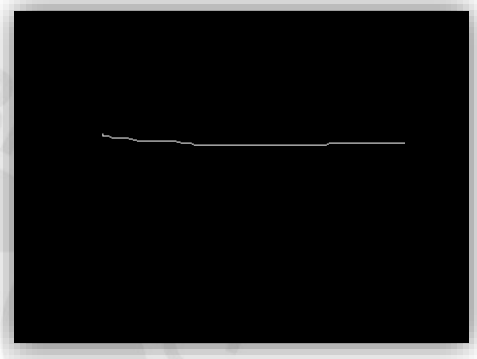
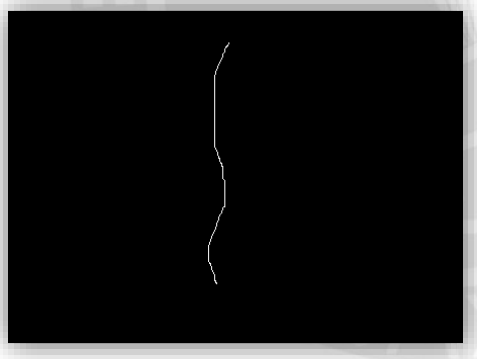
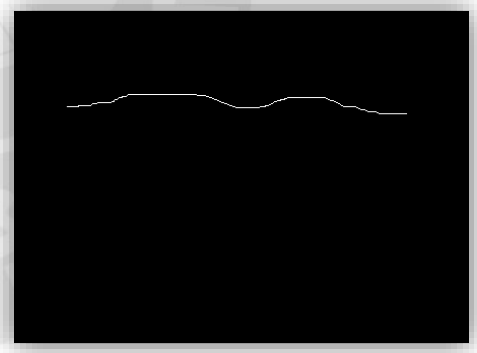
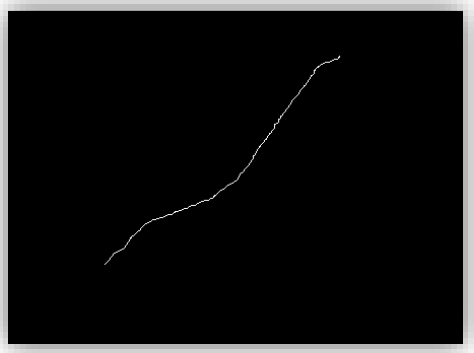
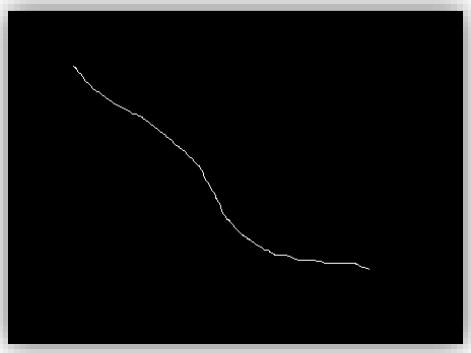
Río meandro 21 	Río meandro 22 
Río meandro 23 	Río meandro 24 
Río meandro 25 	Río meandro 26 
Río meandro 27 	Río meandro 28 

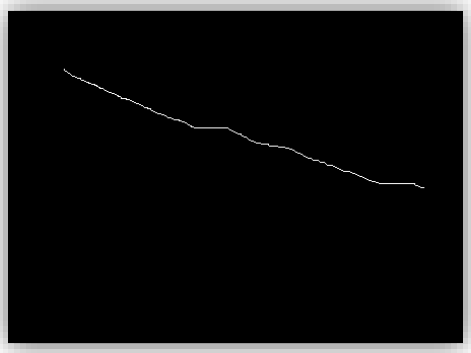
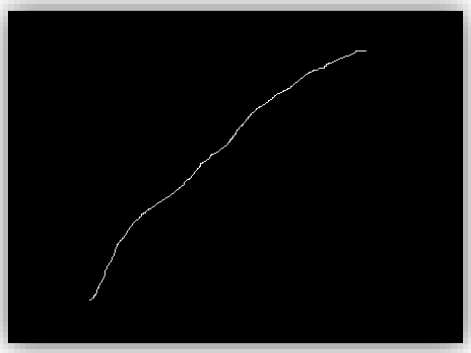
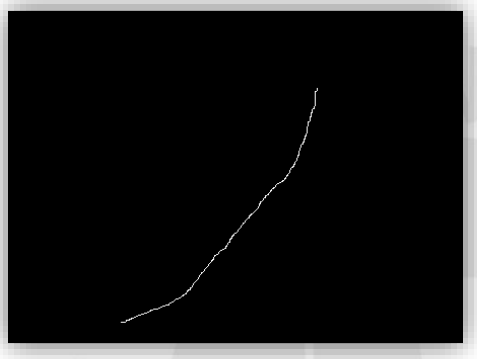
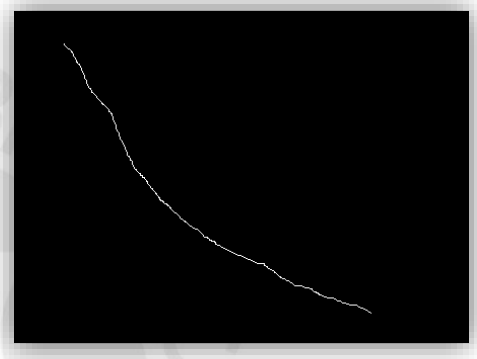
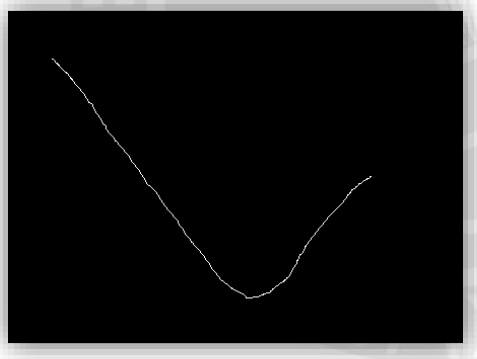
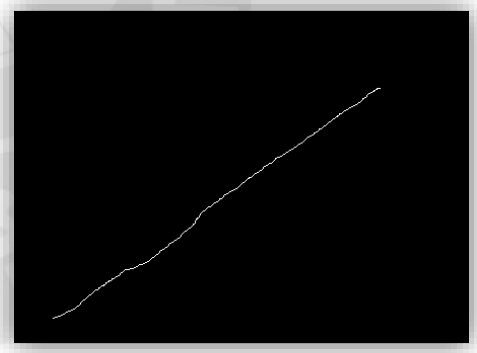
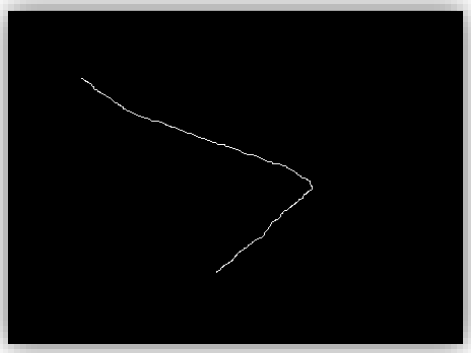
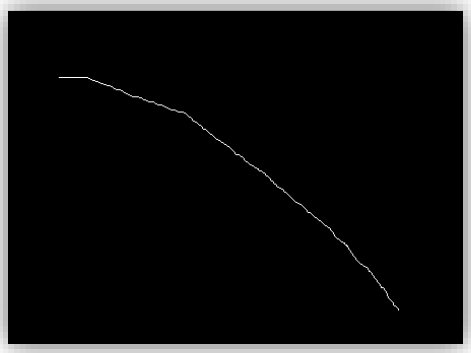
Río meandro 29	Río meandro 30
	
Río meandro 31	Río meandro 32
	
Río meandro 33	Río meandro 34
	
Río meandro 35	Río meandro 36
	

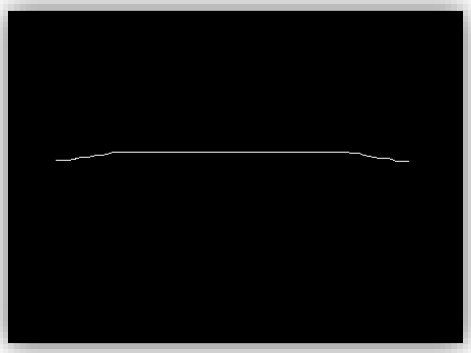
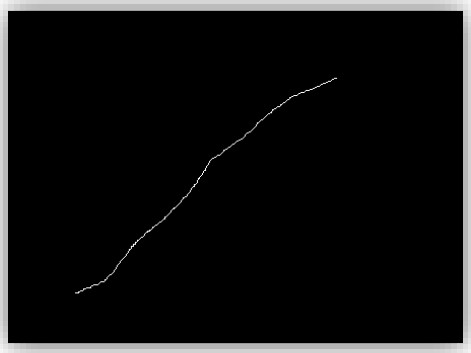
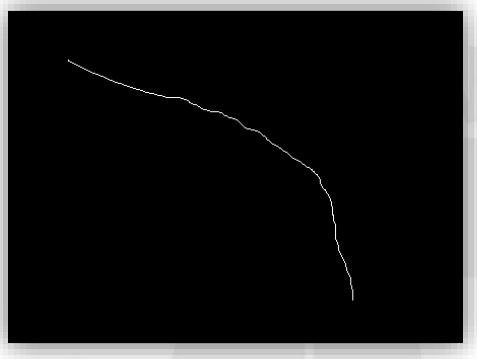
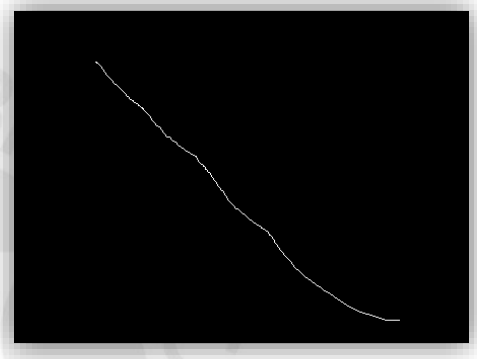
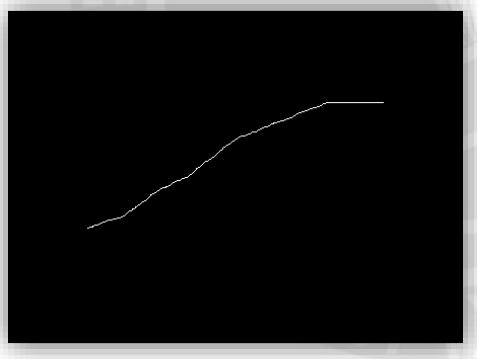
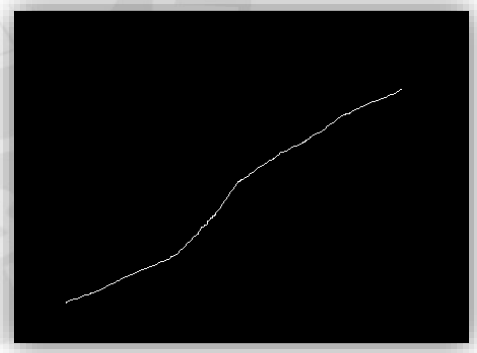
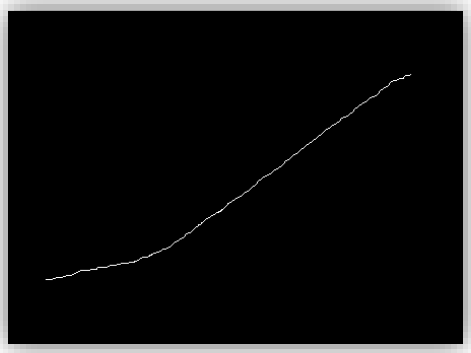
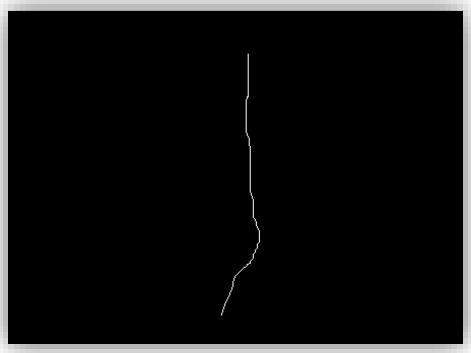
Río meandro 37 	Río meandro 38 
Río meandro 39 	Río meandro 40 

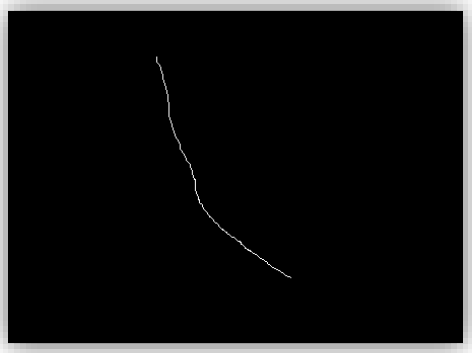
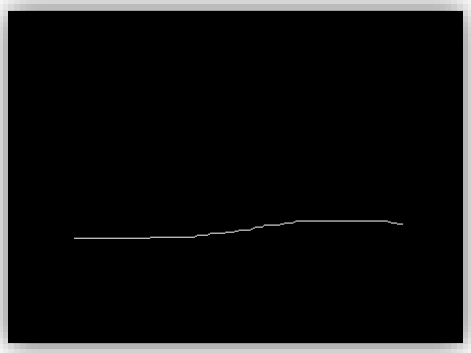
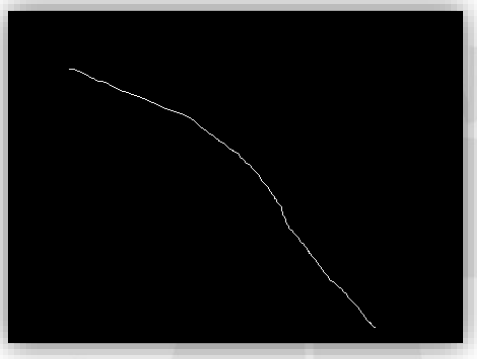
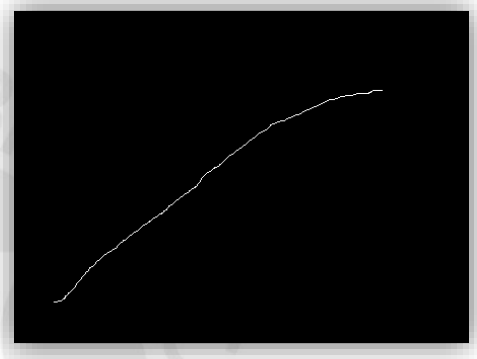
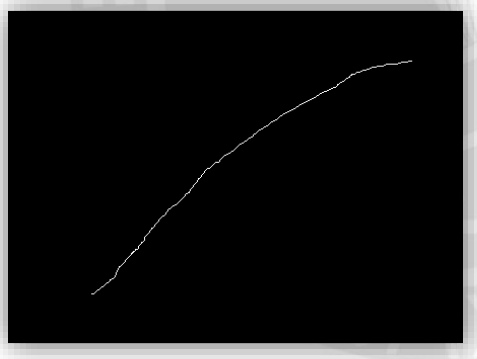
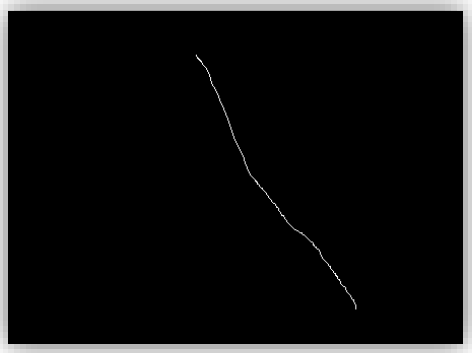
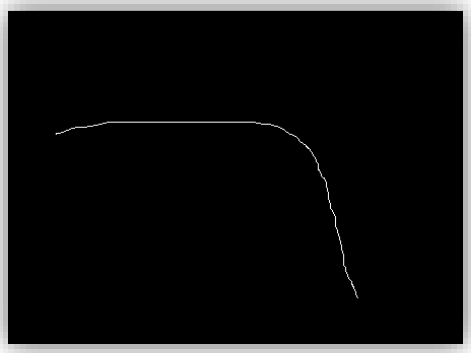
1.7 Esqueletos ríos rectos

Río recto 1 	Río recto 2 
Río recto 3 	Río recto 4 
Río recto 5 	Río recto 6 
Río recto 7 	Río recto 8 

Río recto 9	Río recto 10
	
Río recto 11	Río recto 12
	
Río recto 13	Río recto 14
	
Río recto 15	Río recto 16
	

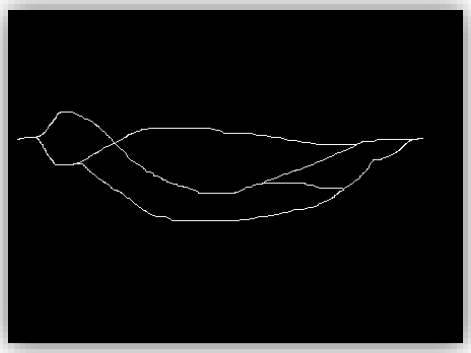
Río recto 17 	Río recto 18 
Río recto 19 	Río recto 20 
Río recto 21 	Río recto 22 
Río recto 23 	Río recto 24 

Río recto 25	Río recto 26
	
Río recto 27	Río recto 28
	
Río recto 29	Río recto 30
	
Río recto 31	Río recto 32
	

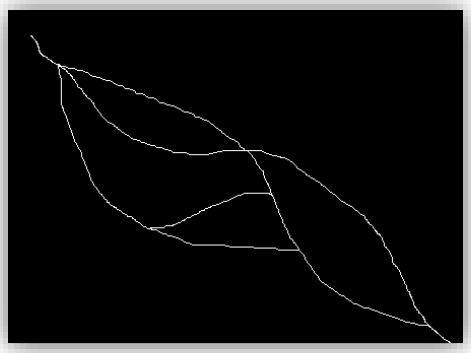
Río recto 33 	Río recto 34 
Río recto 35 	Río recto 36 
Río recto 37 	Río recto 38 
Río recto 39 	Río recto 40 

1.8 Esqueletos ríos trenzados

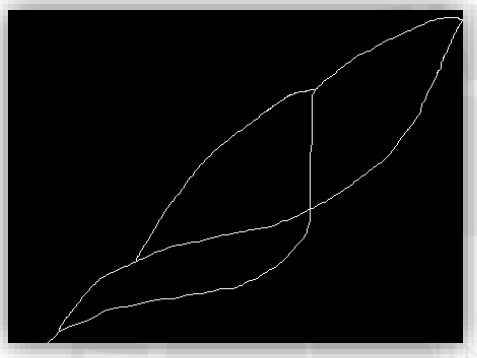
Río trenzado 1



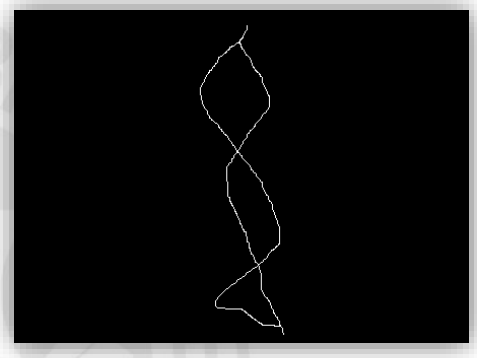
Río trenzado 2



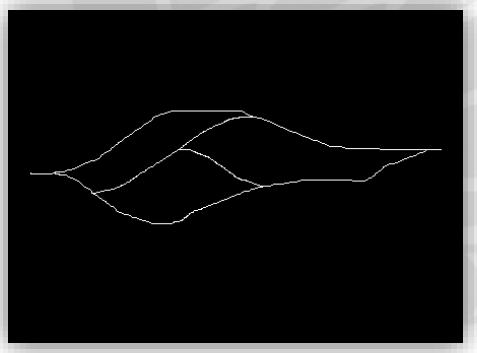
Río trenzado 3



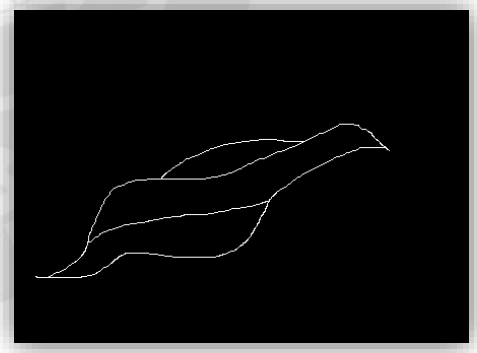
Río trenzado 4



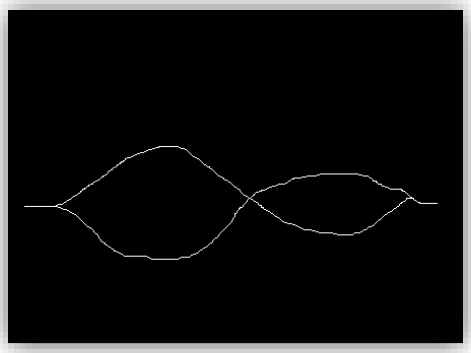
Río trenzado 5



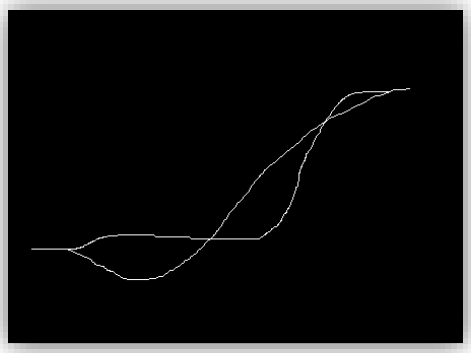
Río trenzado 6

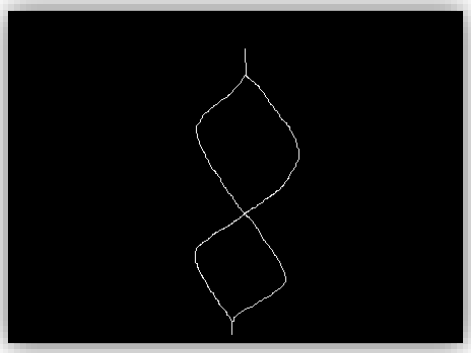
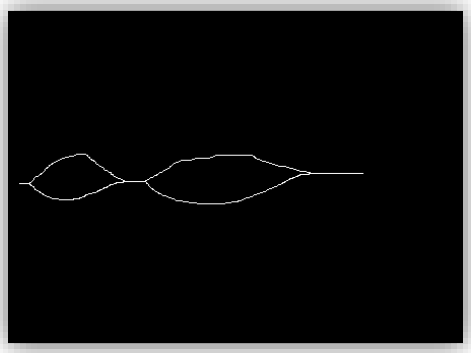
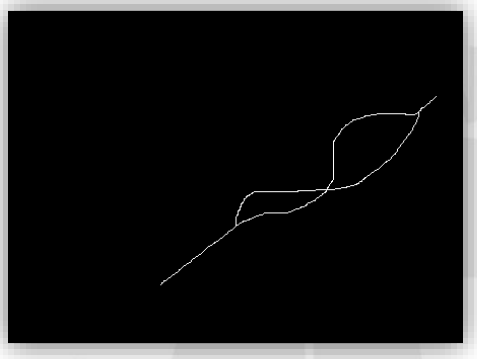
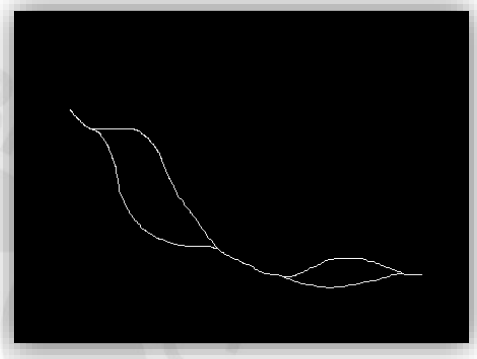
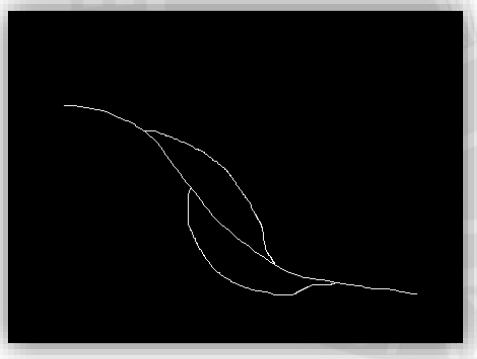
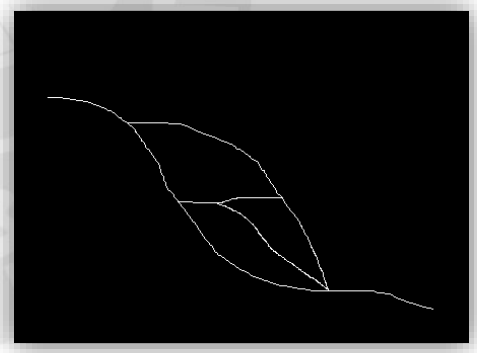
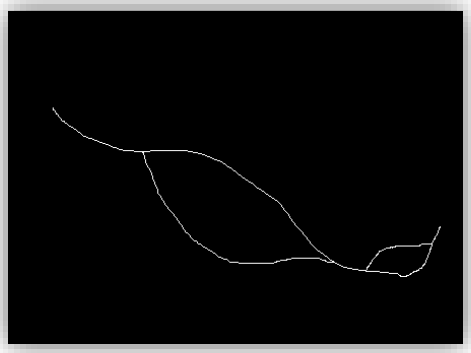
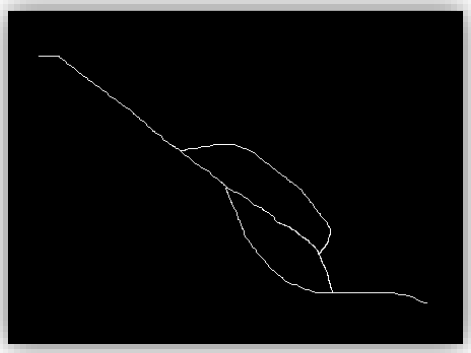


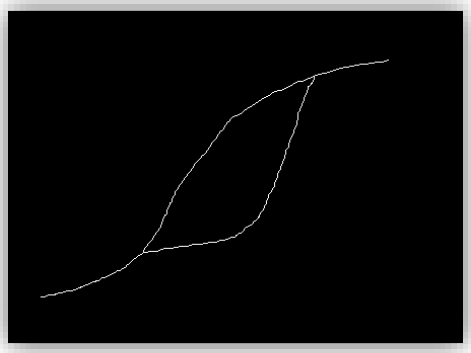
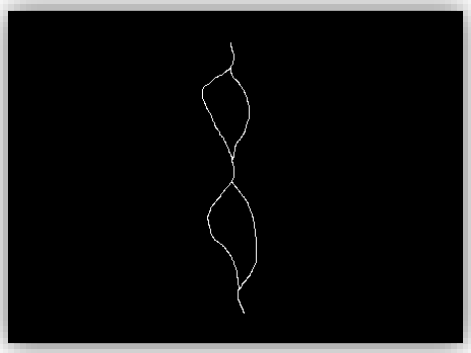
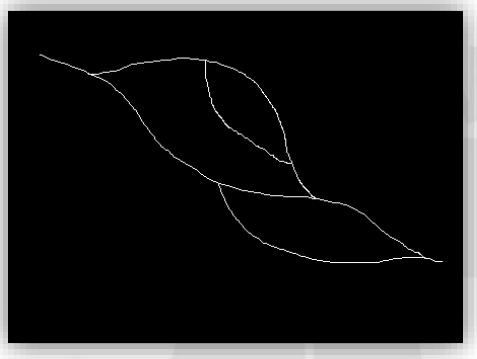
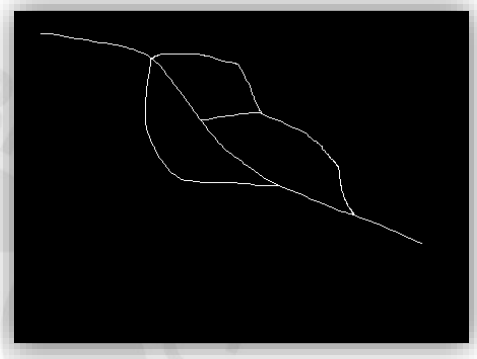
Río trenzado 7



Río trenzado 8

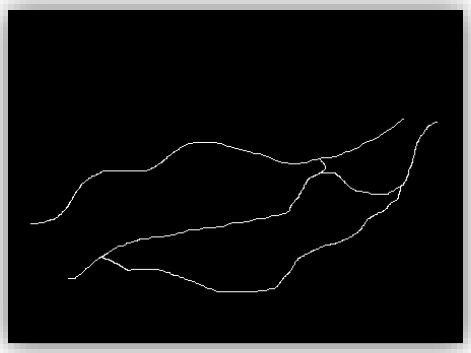


Río trenzado 9	Río trenzado 10
	
Río trenzado 11	Río trenzado 12
	
Río trenzado 13	Río trenzado 14
	
Río trenzado 15	Río trenzado 16
	

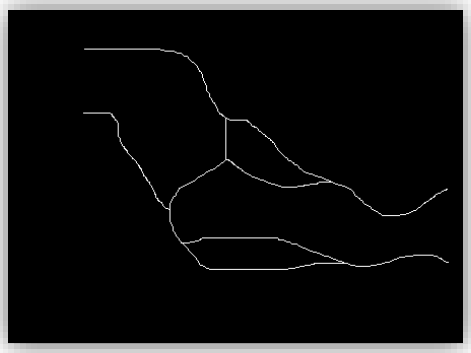
Río trenzado 17	Río trenzado 18
	
Río trenzado 19	Río trenzado 20
	

1.9 Esqueletos ríos anastomosados

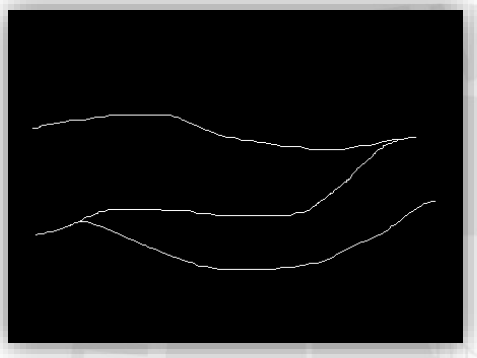
Río anastomosado 1



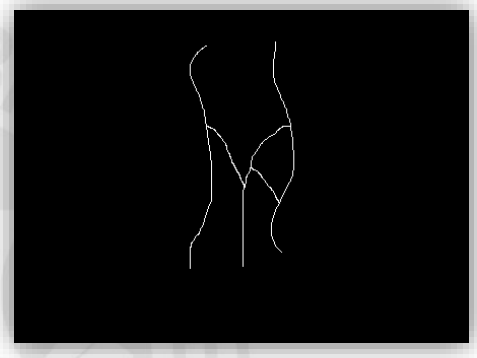
Río anastomosado 2



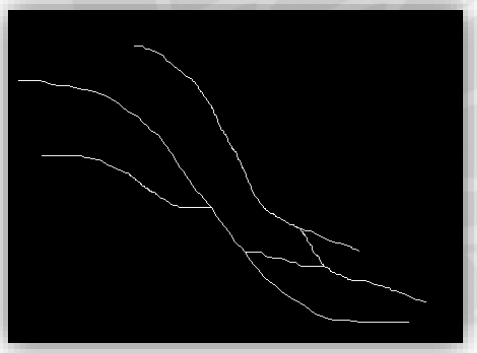
Río anastomosado 3



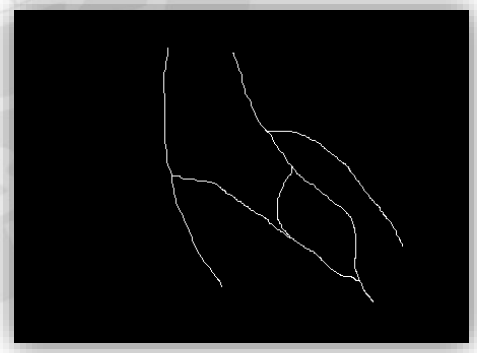
Río anastomosado 4



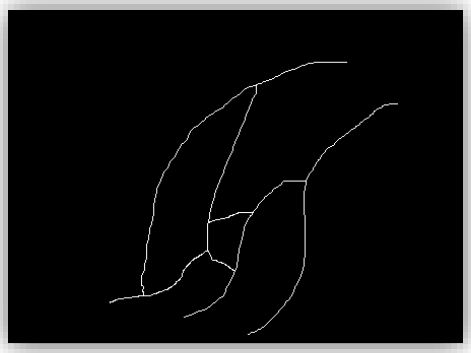
Río anastomosado 5



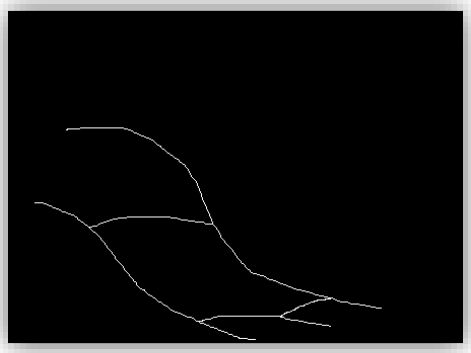
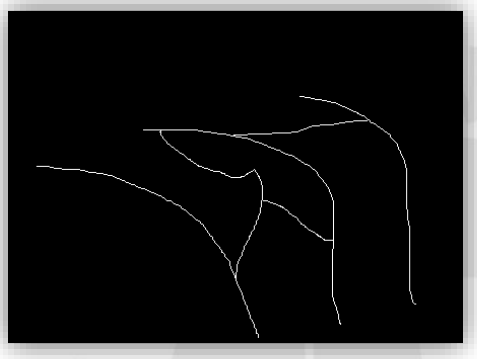
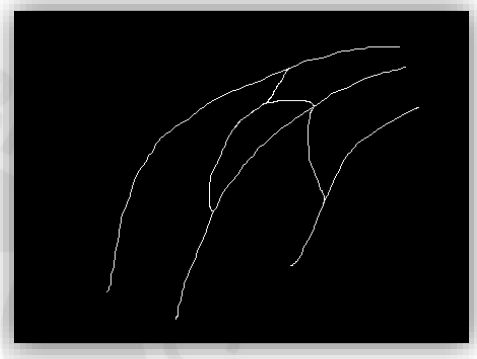
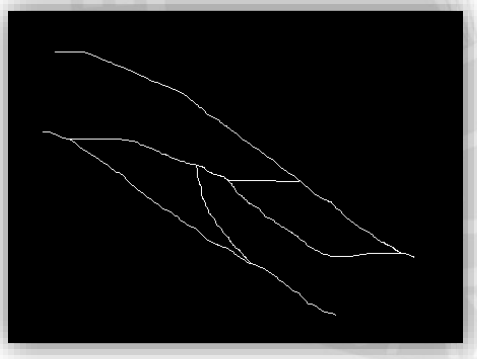
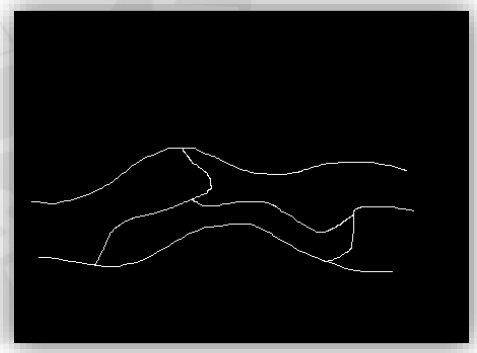
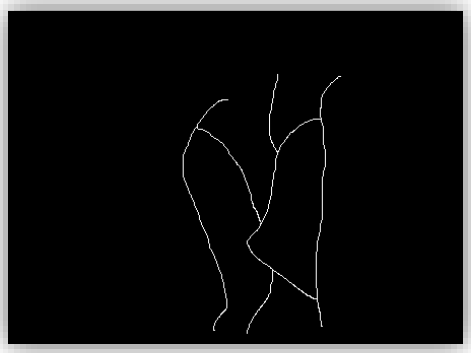
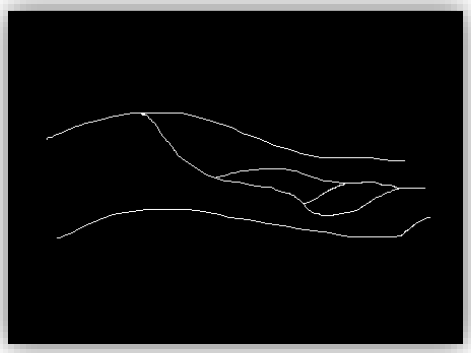
Río anastomosado 6

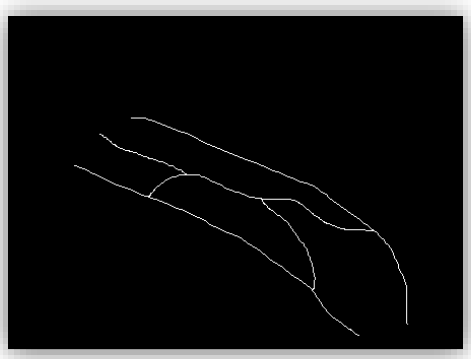
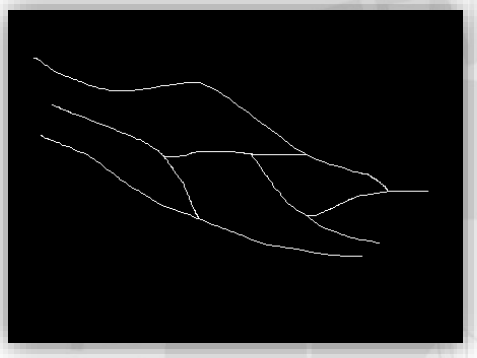
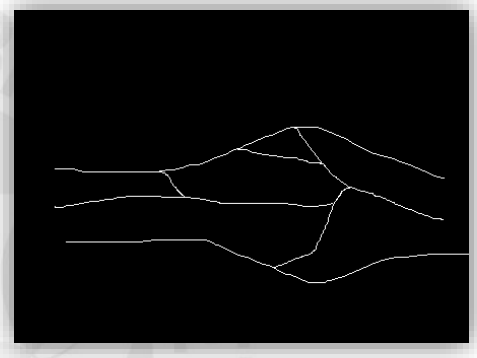


Río anastomosado 7



Río anastomosado 8

Río anastomosado 9	Río anastomosado 10
	
Río anastomosado 11	Río anastomosado 11
	
Río anastomosado 12	Río anastomosado 13
	
Río anastomosado 15	Río anastomosado 16
	

Río anastomosado 17 	Río anastomosado 18 
Río anastomosado 19 	Río anastomosado 20 

2 Curvas ROC

2.1 Prueba 1 – SVM uno contra uno

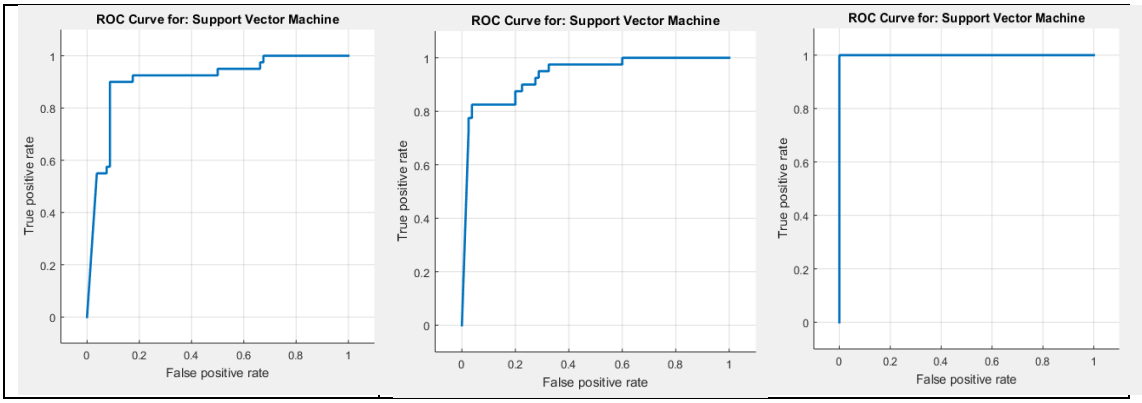


Figura 2.1. Curvas ROC prueba 1- SVM uno contra uno
Clases positivas por columna respectivamente: meandros, rectos, multicanal
Elaboración propia

2.2 Prueba 1 – SVM uno contra el resto

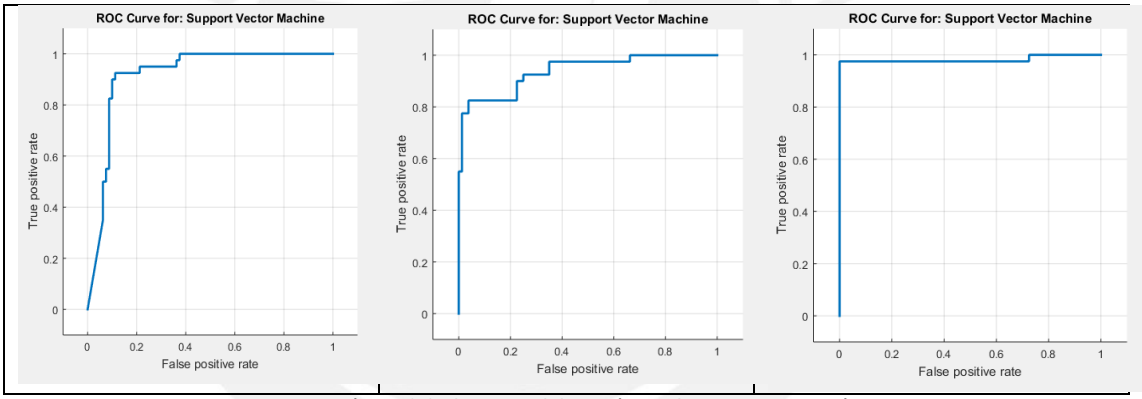


Figura 2.2. Curvas ROC prueba 1- SVM uno contra el resto
Clases positivas por columna respectivamente: meandros, rectos, multicanal
Elaboración propia

2.3 Prueba 1 – Adaboost con 200 iteraciones

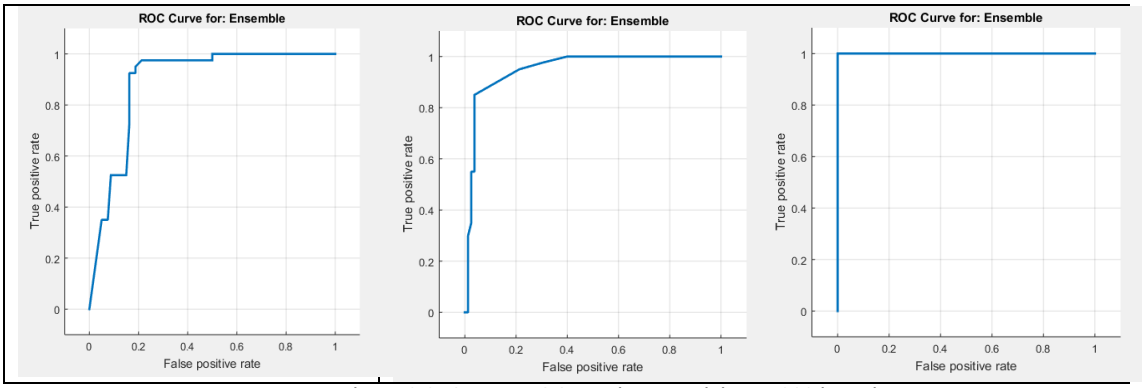


Figura 2.3. Curvas ROC prueba 1 – Adaboost 200 iteraciones

Clases positivas por columna respectivamente: meandros, rectos, multicanal

Elaboración propia

2.4 Prueba 1 – Adaboost con 1000 iteraciones

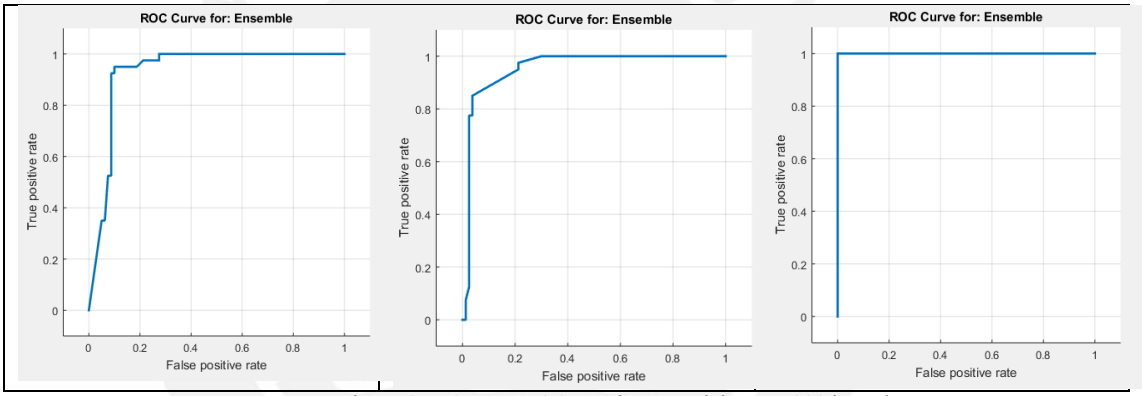


Figura 2.4. Curvas ROC prueba 1 – Adaboost 1000 iteraciones

Clases positivas por columna respectivamente: meandros, rectos, multicanal

Elaboración propia

2.5 Prueba 2 – SVM uno contra uno

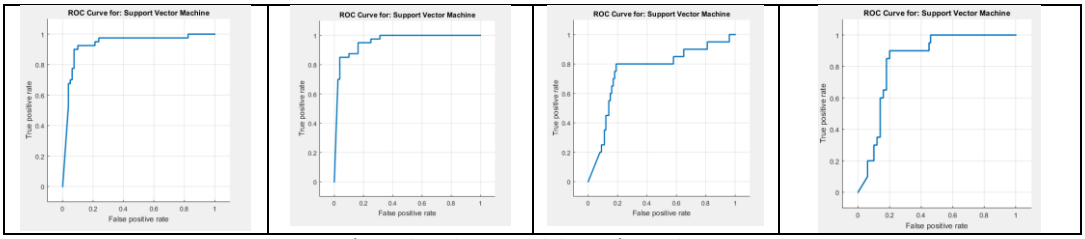


Figura 2.5. Curvas ROC prueba 1- SVM uno contra uno

Clases positivas por columna respectivamente: meandros, rectos, trenzados, anastomosados

Elaboración propia

2.6 Prueba 2 – SVM uno contra el resto

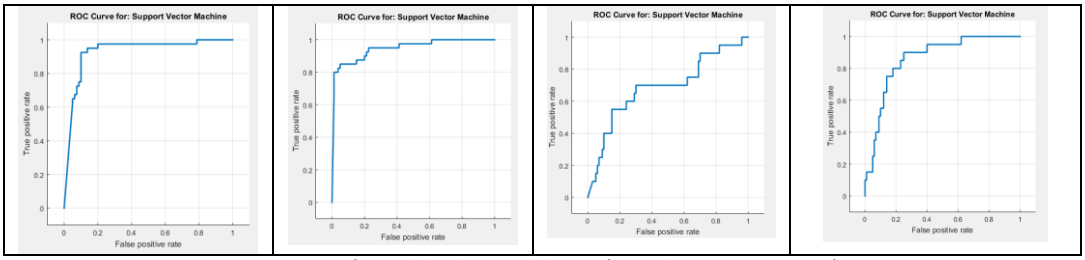


Figura 2.6. Curva ROC prueba 1- SVM uno contra el resto
Clases positivas por columna respectivamente: meandros, rectos, trenzados, anastomosados
Elaboración propia

2.7 Prueba 2 – Adaboost con 200 iteraciones

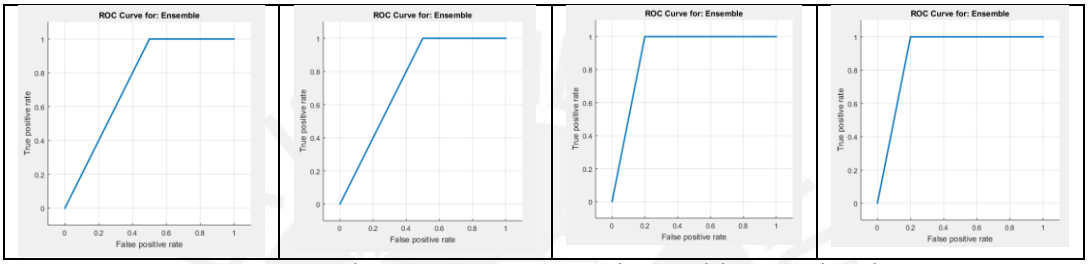


Figura 2.7. Curvas ROC prueba 2 – Adaboost 200 iteraciones
Clases positivas por columna respectivamente: meandros, rectos, trenzados, anastomosados
Elaboración propia

2.8 Prueba 2 – Adaboost con 1000 iteraciones

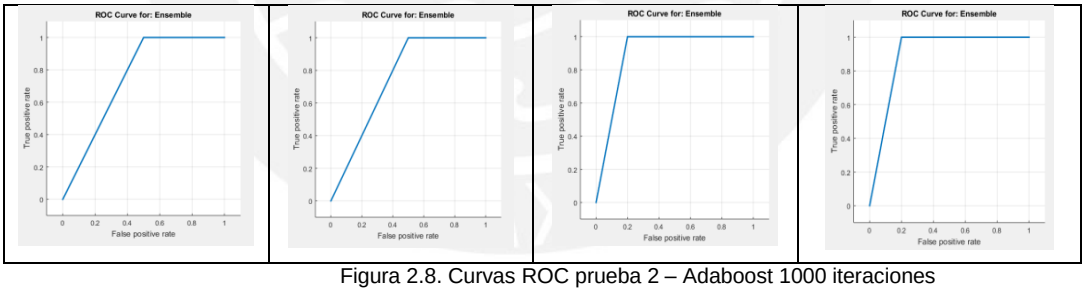


Figura 2.8. Curvas ROC prueba 2 – Adaboost 1000 iteraciones
Clases positivas por columna respectivamente: meandros, rectos, trenzados, anastomosados
Elaboración propia

2.9 Prueba 3 – MultiClassClassifier

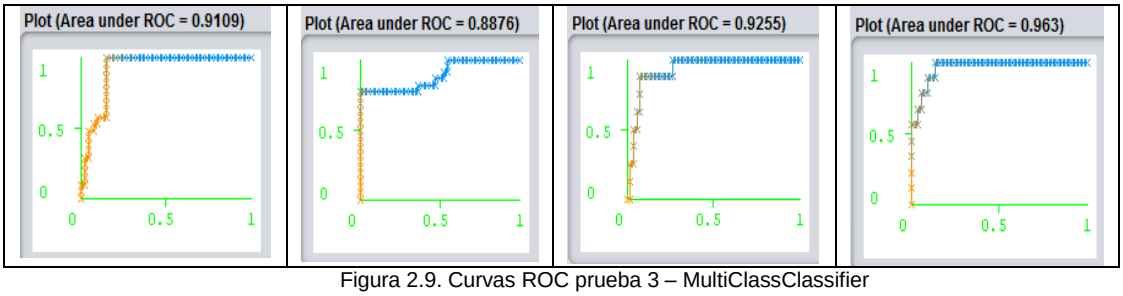


Figura 2.9. Curvas ROC prueba 3 – MultiClassClassifier

Clases positivas por columna respectivamente: meandros, rectos, trenzados, anastomosados

Elaboración propia