Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Dimitri Leca **Computer Vision: Lab 5** **30/11/2015**

Maher Nadar **Thinning**

NOTE: Kindly find attached the code of this practice in order to verify our work

1) zhang-suen algorithm implementation

In this part of the practice, we have to implement a previously established algorithm developed by **T. Y. ZHANG and C. Y. SUEN** in order to get the skeleton shape of a particular object. The algorithm consists of 2 sub iterations:

1- Deleting the south-east boundary points and the north-west corner points.

2- Deleting the north-west boundary points and the south-east corner points.

The endpoints and the pixel connectivity are preserved, so as to end up with a single pixel wide skeleton of the original project.

P_9 $(i-1, j-1)$	P_2 $(i-1, j)$	P_3 $(i-1, j+1)$
P_8 $(i, j-1)$	P_1 (i, j)	P_4 $(i, j+1)$
P_7 $(i+1, j-1)$	P_6 $(i+1, j)$	P_5 $(i+1, j+1)$

FIGURE 1. Designations of the nine pixels in a 3 × 3 window.

Considering the pixel in question while going through the image to be P1 in the above figure, we can set the 3x3 matrix containing it along with its surrounding pixels as in the Figure (respecting the same order displayed in the figure).

1- In order to satisfy the first condition introduced above, we search for the pixels that satisfy the following:

$$(a)\ \ 2 \leq B(P_1) \leq 6$$

$$(b)\ \ A(P_1) = 1$$

$$(c)\ \ P_2 * P_4 * P_6 = 0$$

$$(d)\ \ P_4 * P_6 * P_8 = 0$$

Where B (P1) in a) signifies the binary sum of the elements P2 to P9;
A (P1) in b) is the number of 01 patterns in the ordered set P1 to P9;

2- In order to satisfy the second condition, we search for the pixels that satisfy the following:
   a) And b) points remain the same;

$$(c') \quad P_2 * P_4 * P_8 = 0$$

$$(d') \quad P_2 * P_6 * P_8 = 0$$

Taking these sub iterations as our building blocks, we can now set-up our algorithm according to the following pseudo-code:

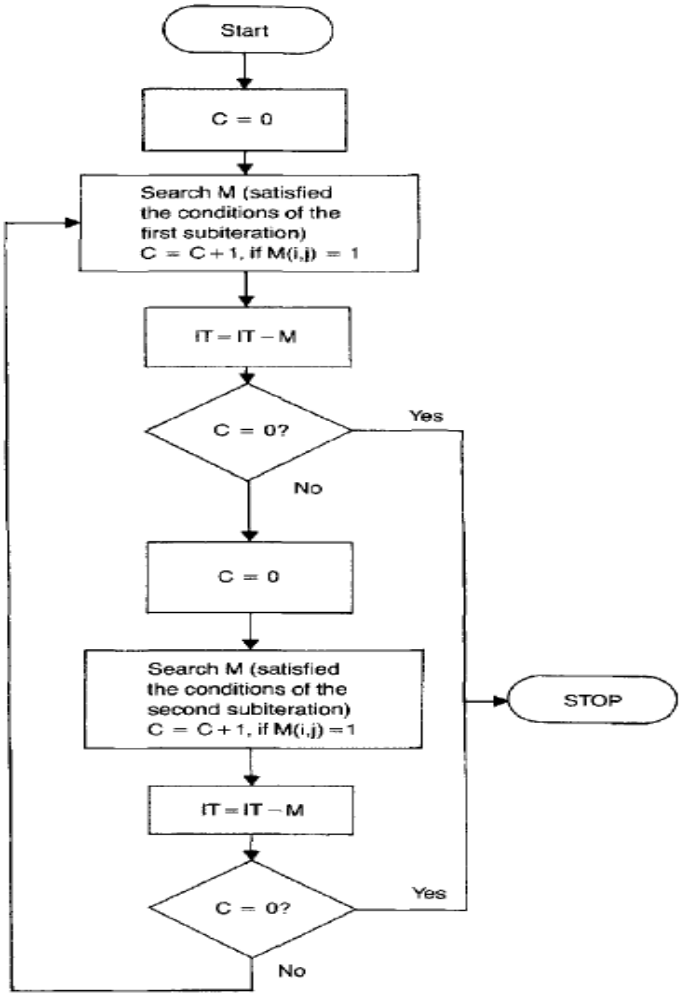


FIGURE 6. Flowchart of the thinning algorithm.

Main Coding chunks

Code of the main image searching:

```
% We initialize the values:

IT = img;
C = 0;

algo_finished = false;

sizex = size(IT,1);
sizey = size(IT,2);


while 1==1

    % We search M ( first sub iteration)
    M = zeros(sizex, sizey);

    for I = 2:sizex-1
        for j = 2:sizey-1
            M(I,j) = checkFirstSubiteration(IT, I, j);
        end
    end
    C = sum(sum(M));


    IT = IT - M;

    if C == 0
        break

    else % second sub iteration

            % We search M ( first iteration)
            M = zeros(sizex, sizey);

            for I = 2:sizex-1
                for j = 2:sizey-1
                    M(I,j) = checkSecondSubiteration(IT, I, j);
                end
            end
             C = sum(sum(M));

            IT = IT - M;


            if C == 0
                break
            end
    end

    end
```

Code of 1st sub iteration:

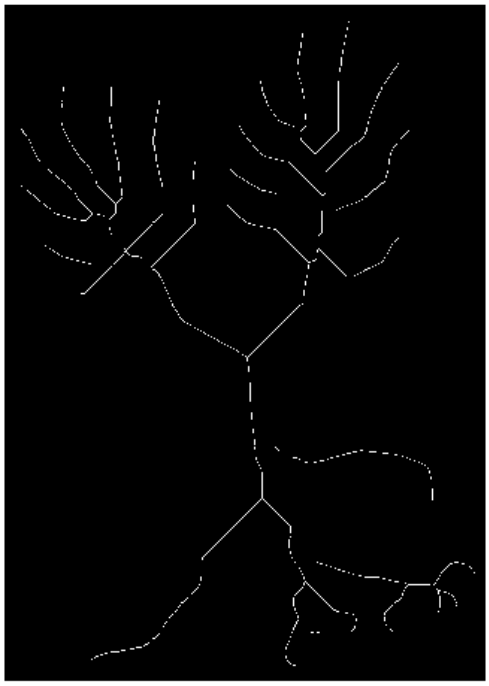
```
function output_int = checkFirstSubiteration(K, i, j)


P1 = K(i,j);
P2 = K(i-1,j);
P3 = K(i-1,j+1);
P4 = K(i,j+1);
P5 = K(i+1,j+1);
P6 = K(i+1,j);
P7 = K(i+1,j-1);
P8 = K(i,j-1);
P9 = K(i-1,j-1);

% We compute A.

A = 0;
if (P2 == 0) && (P3 == 1)
    A = A+1;
end
if (P3 == 0) && (P4 == 1)
    A = A+1;
end
if (P4 == 0) && (P5 == 1)
    A = A+1;
end
if (P5 == 0) && (P6 == 1)
    A = A+1;
end
if (P6 == 0) && (P7 == 1)
    A = A+1;
end
if (P7 == 0) && (P8 == 1)
    A = A+1;
end
if (P8 == 0) && (P9 == 1)
    A = A+1;
end
if (P9 == 0) && (P2 == 1)
     A = A+1;
end


% We compute B.
B = P2 + P3 + P4 + P5 + P6 + P7 + P8 + P9;

    if (2 <= B) && (B <= 6) && (P2*P4*P6 == 0) && (P4*P6*P8 == 0) && (A == 1) &&
(P1 == 1)
        output_int = 1; % Conditions satisfied
    else
        output_int = 0;
    end

end
```

Code of 2nd sub iteration:

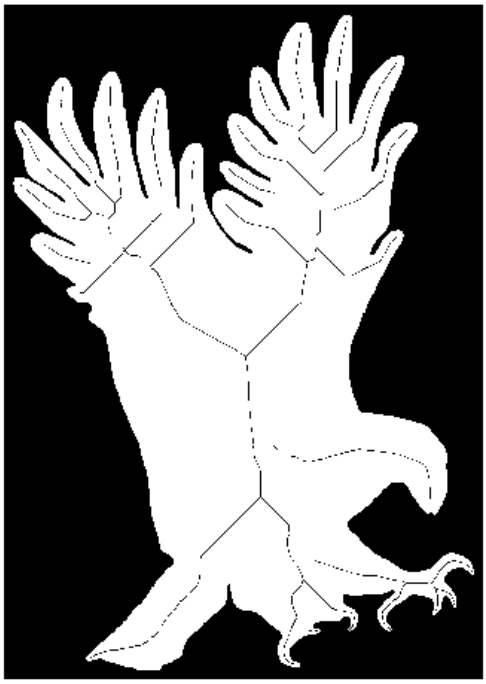
```
function output_int = checkSecondSubiteration(K, i, j)

P1 = K(i,j);
P2 = K(i-1,j);
P3 = K(i-1,j+1);
P4 = K(i,j+1);
P5 = K(i+1,j+1);
P6 = K(i+1,j);
P7 = K(i+1,j-1);
P8 = K(i,j-1);
P9 = K(i-1,j-1);

A = 0;

if (P2 == 0) && (P3 == 1)
    A = A+1;
    end
    if (P3 == 0) && (P4 == 1)
        A = A+1;
    end
    if (P4 == 0) && (P5 == 1)
        A = A+1;
    end
    if (P5 == 0) && (P6 == 1)
        A = A+1;
    end
    if (P6 == 0) && (P7 == 1)
        A = A+1;
    end
    if (P7 == 0) && (P8 == 1)
        A = A+1;
    end
    if (P8 == 0) && (P9 == 1)
        A = A+1;
    end
    if (P9 == 0) && (P2 == 1)
        A = A+1;
    end


B = P2 + P3 + P4 + P5 + P6 + P7 + P8 + P9;



if (2 <= B) && (B <= 6) && (P2*P4*P8 == 0) && (P2*P6*P8 == 0) && (A == 1) && (P1 == 1)


    output_int = 1; % Conditions satisfied
else
    output_int = 0;
end

end
```

Results

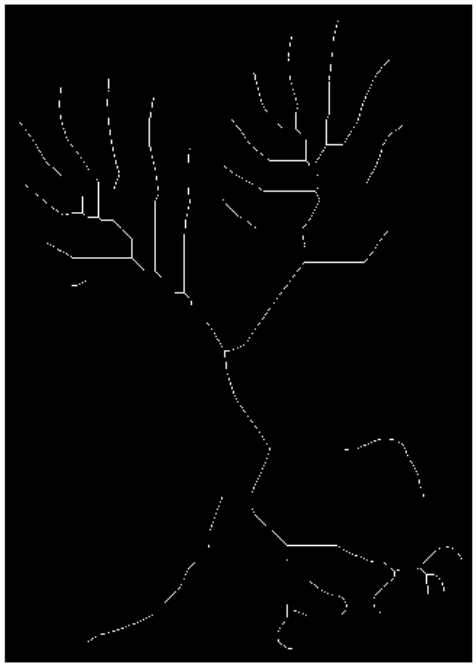
- ***Flying-Eagle example***



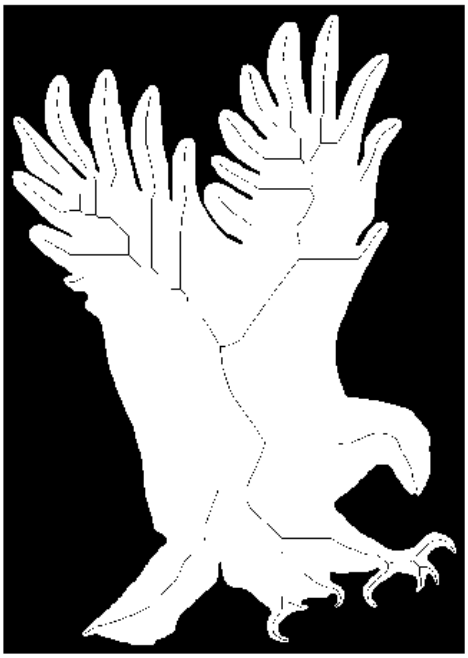
Eagle Skeleton **zhang-suen**



Eagle Skeleton **zhang-suen** inside the original

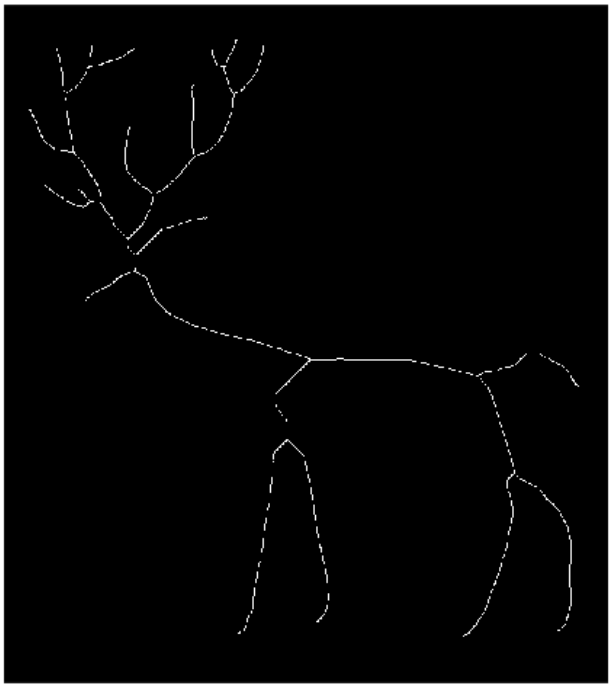


Eagle Skeleton **bwmorph**

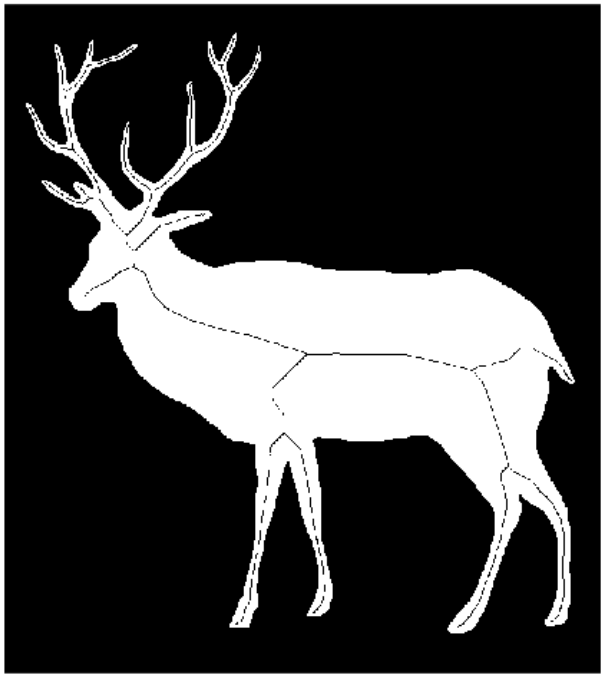


Eagle Skeleton **bwmorph** inside the original

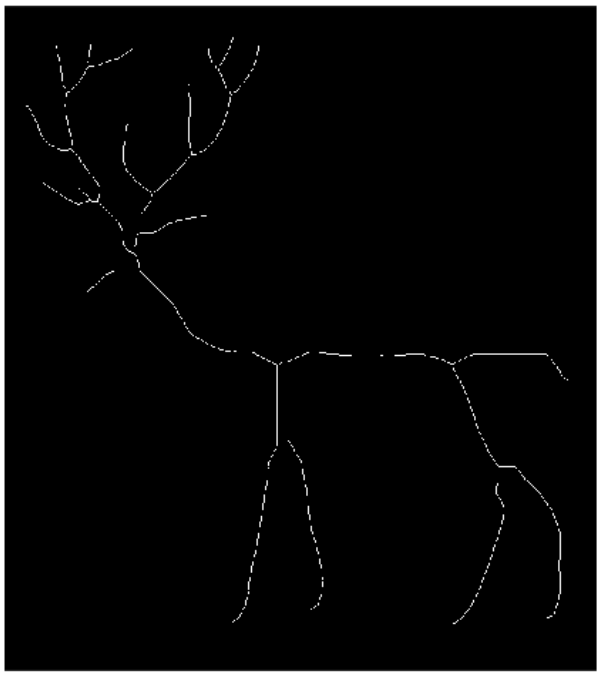
- ***Deer example:***



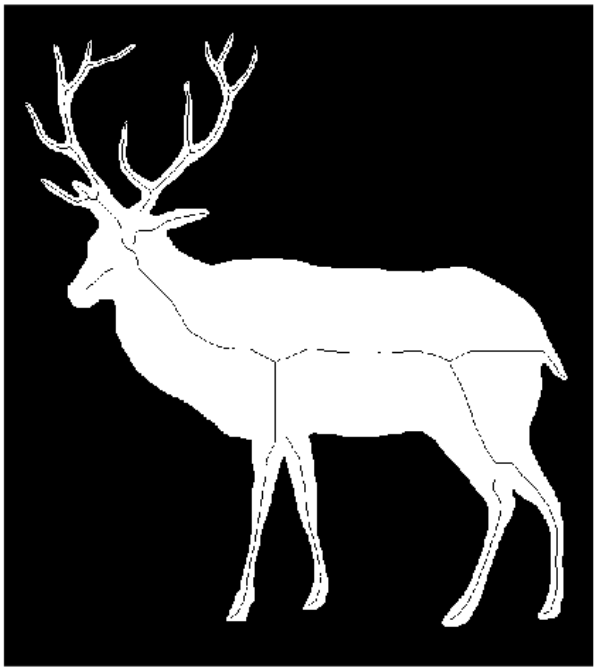
Deer Skeleton **zhang-suen**



Deer Skeleton **zhang-suen** inside the original

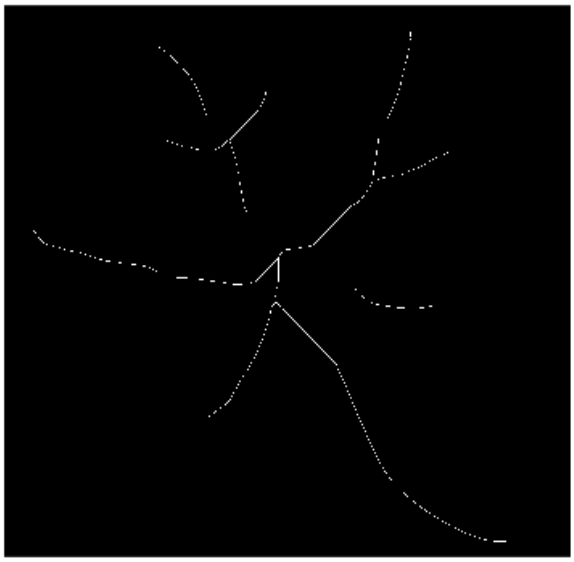


Deer Skeleton **bwmorph**

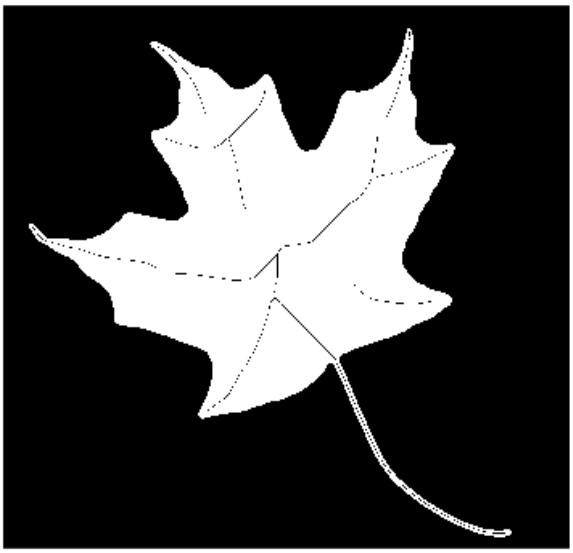


Deer Skeleton **bwmorph** inside the original

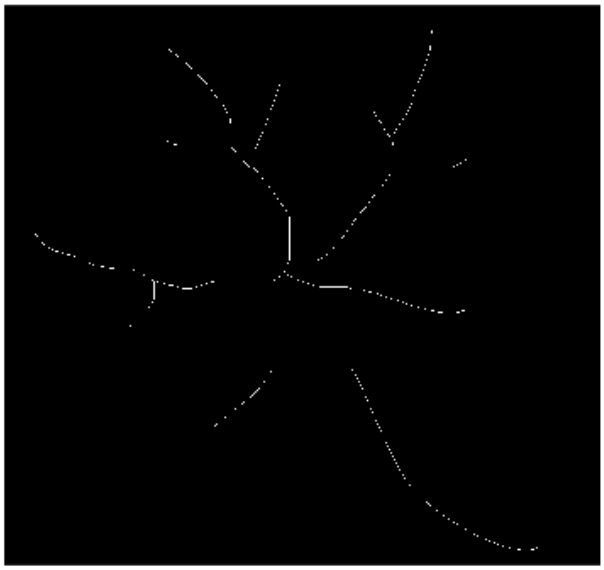
- ***Leaf example***



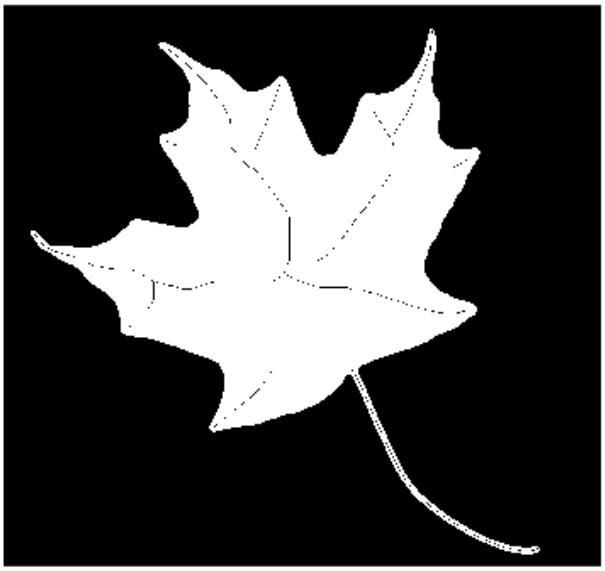
Leaf Skeleton **zhang-suen**



Leaf Skeleton **zhang-suen** inside the original



Leaf Skeleton **bwmorph**



Leaf Skeleton **bwmorph** inside the original

Comments and analysis

As can be observed in the above displayed pictures (Although the resolution of the figures is not quite high enough for easy discerning of outputs), the 2 methods reach comparable results. Nonetheless, the final skeleton obtain in the 1st method can be seen to have somewhat different branching as the second method.

Another comparative criterion one should consider here is the elapsed time needed in order to achieve the skeleton in the different techniques. As can be seen in the table below, the bwmorph() function in Matlab is accomplishing the task in a much faster manner:

Elapsed Time	bwmorph (s)	Zhang Swen (s)
Eagle	0.136841	54.7364
Deer	0.068075	27.23
Leaf	0.457422	182.9688

END