Skeletonization, skeleton pruning and simple skeleton graph construction example in Matlab.

Andoni Beristain Iraola

beristainandoni@yahoo.es

April 2009

Contents

1. Introduction

This document describes an add-on for the Matlab implementation of the skeleton pruning algorithm in references 2 and 3. This add-on computes an adjacency matrix, a way to represent a graph, based in the article in reference 1, and shows it graphically.

This list of files compute the skeleton using the distance transform and prune the skeleton using the reference 2. These files have been downloaded from reference 4.

- checkskeleton1.m
- curvediv1.m
- div_skeleton_new.m
- evolution.m
- findConvex.m
- GetContour1.m
- Intersecto.m
- Lab2Pos.m
- MarkOther.m
- pathDFS1.m
- SkeletonGrow1.m

This is the list of files implementing the skeleton to graph transformation based on reference 1:

- cListaNodos.m – Graph node array class. Uses nodo.m.
- limpiarNodosFp.m – Function to refine initial graph node list.
- nodo.m – Graph node class.
- skel2Graph.m – Function to obtain the skeleton graph.
- Test1Elemento.m – Example procedure showing how to use skel2Graph function.
- vCaracV1.m – Adjacency matrix construction function.

2. Description

In this section the contents and functionality of each developed file is commented.

Test1elemento is the main procedure. It calls to skel2Graph which in turn calls to limpiarNodosFP and vCaracV1.

a. Test1elemento:

This is the main function file. Call this file to execute the example.

This file loads a binary image file, which path must be typed in the command window, between ' characters. The path can be relative or absolute. Example: 'c:\sampleImage.bmp' . Then computes its pruned skeleton and finally obtains the skeleton graph.

The function first makes a call to `div_skeleton_new` which obtains a pruned skeleton. Then calls `skel2Graph` to obtain the graph. And finally shows a graphical representation of the adjacency matrix where each node is labelled with its id and surrounded by a circle.

b. Skel2Graph:

This function obtains the adjacency matrix and final node list, as an object of the class cListaNodos. It is a simplified version of the algorithm proposed at reference 1.

Function input:

- imEsqueleto: Pruned skeleton image.

Function output:

- listaNodos: cListaNodos class object with the final nodes.
- matrizAdyacencia: Adjacency matrix representing the output graph.

Steps:

1. Obtain an initial classification of skeleton pixels into:
    a. BRANCH_POINT: pixels of a skeleton branch.
    b. END_POINT: pixels at the end of a skeleton branch.
    c. FORK_POINT: pixels connecting several skeleton branches on it.
2. Clean the FORK_POINTs, calling to limpiarNodosFp.
3. Show the results:
    a. Feature point group labeling. Each colored blob represents a different virtual or real node of the graph. If the node has 1 pixel it is real and if it is formed by several pixels, then its centroid is considered and it is treated as only one virtual node.
    b. Skeleton with final nodes. Skeleton with an overlay showing the fork points and the starting of their associated branches.

4. Obtain the adjacency matrix calling to vCaracV1.

c. limpiarNodosFp

This function refines the initial FORK_POINT list, to correct several skeletonization errors. Some FORK_POINTS will become BRANCH_POINTS and others that are connected, forming groups will be replaced by a virtual point with the sum of all the branches connected to every pixel in the group and centroid of the group as position.

Function input:

- imClasif: image showing the type of skeleton point for each pixel (BRANCH_POINT, END_POINT, FORK_POINT).
- lFork_: x and y coordinate of the initial FORK_POINT list.
- lEnd_: x and y coordinate of the initial END_POINT list.

Function output:

- listaNodos: cListaNodos Class object containing the final nodes.
- imGroups: Node labelling image. For each skeleton pixel indicates the node id it belongs to. 0 means that it doesn't belong to any node.

Steps:

1. Create an image with only the fork points.
2. Use connected component labelling technique to assign a label to each group.
3. Obtain all the real and virtual nodes computing the centroid and branch connecting points for each labelled component.
4. Add all the end points as nodes to the list with incrementing id.

d. vCaracV1

Based on the node classification of limpiarNodosFP, obtains the adjacency matrix and the final node list. This version employs only the feature points and the links between them. Skeleton branches are not characterized in any way, but the actual code implementation makes easy to add this information.

Function input:

- iBranch: Binary image. If iBranch(x,y) is BRANCH_POINT then its value is 1, 0 otherwise.

- iGroups: Binary image. If iGroups(x,y) is not a node pixel then its value is 0, otherwise indicates the node id to which it belongs to
- lNodes: cListaNodos object containing the graph node list.

Function output:

- mAdjacency: Adjacency matrix of the resulting skeleton graph.
- nNodes: cListaNodos object with the resulting node list.

Steps:

1. For each node travel all over its neighbouring branches until finding another node.
2. Mark the relations in the adjacency matrix.
3. Remove incorrectly labelled nodes.

e. cListaNodos

Dynamic array of nodo class objects. Provides addition, deletion and access to the elements. To minimize spatial requirements, it is implemented as pointer.

Properties:

- lista: node list.
- numElem: number of nodes.

Functions:

- Function nod=n(this,i): returns a copy of the node at index i.
- Function addN(this,i): adds a node to the end of the list.
- Function delN(this,i): removes a node from the end of the list.
- Function total=numNodos(this): returns the number of nodes in the list.

f. nodo

This class contains all the information of each graph node.

Properties:

- x, y: coordinates of the node.
- Tipo: node type: END_POINT, FORK_POINT or BRANCH_POINT.

- esAbstracto: if false, the node represents only one skeleton point, otherwise represents a virtual node, composed of several adjacent points.
- lVecRama_: coordinates of the first skeleton BRANCH_POINT that originates from the node for each emanating branch.
- numVecRama: number of branches originating in the node.

Functions:

- function anadirVecino(this,x,y): Adds a neighbouring branch's initial pixel.
- function val=numVecinos(this): Returns the number of neighbours.
- Function [x,y]=vecinoI(this,i): Returns the coordinates of the first BRANCH_POINT for the i-th neighbouring branch.
- Function eliminarVecino(this,i): Deletes i-th neighbour of the node.

5. Usage

For testing purposes, call Test1elemento and write a binary image path between ' characters in the command window. Both relative and absolute paths are allowed.

To use the graph computation procedure, call to skel2Graph with the appropriate parameters.

6. References

1. Liu, Ke and Huang,, Yea C. and Suen,, Ching Y. Identification of Fork Points on the Skeletons of Handwritten Chinese Characters. IEEE Trans. Pattern Anal. Mach. Intell. 21(10) 1999. Pp. 1095—1100. IEEE Computer Society (Washington, DC, USA).
2. Bai, Xiang and Latecki, Longin J. and Liu, Wen Y. Skeleton Pruning by Contour Partitioning with Discrete Curve Evolution. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(3) 2007. Pp. 449--462. IEEE Computer Society (Washington, DC, USA).
3. http://knight.cis.temple.edu/~shape/partshape/structure/
4. http://www.cis.temple.edu/~latecki/Programs/skeletonPruning07.htm

7. Acknowledgments

We appreciate the help of Xiang Bai for letting us use his code as base to illustrate this procedure.