Daily Thesis-work Report

Javier Fernández

April 20, 2010

1 General topics addressed

- Distance transform approaches (research)

2 On distance transformation

Metric: the distance between two points is the sum of the (absolute) differences of their coordinates. The taxicab metric is also known as rectilinear distance, L1 distance or $\ell 1$ norm (see Lp space), city block distance, Manhattan distance, or Manhattan length, with corresponding variations in the name of the geometry. `http://en.wikipedia.org/wiki/Taxicab_geometry`

Different Distance metric: `http://www.mathworks.com/access/helpdesk_r13/help/toolbox/images/morph15.html`
`http://www.mathworks.com/access/helpdesk/help/toolbox/images/bwdist.html`

City block: The city block distance metric measures the path between the pixels based on a 4-connected neighborhood. Pixels whose edges touch are 1 unit apart; pixels diagonally touching are 2 units apart.

Chessboard: The chessboard distance metric measures the path between the pixels based on an 8-connected neighborhood. Pixels whose edges or corners touch are 1 unit apart.

3 Fast Distance transform in two scans using 3x3

Distance transformation (DT) is to convert a digital binary image that consists of object (foreground) and non-object (background) pixels into another image in which each object pixel has a value corresponding to the minimum distance from the background by a distance function

NICE papers to take theoretical documentation

3.1 Implementation of Manhattan and Chessboard distance transformation

Following the procedure described on the paper it was donde a two-scan process to fill the distance matrix. The first one is done from left to right, top to bottom, (forward scan). The second one, backward scan, is done from right to left, bottom to top. The neighbors are calculated depending on the distance metric. The corresponding neighborhoods N_1 and N_2 are used in each scan respectively.

The obtained result give an aceptable distance transformation. But looks like it needs some overwork to find the skeleton appropriately.

- Interesting disertion about why one or the other is good at: **Ridge points in euclidean distance maps**

3.2 Euclidean distance transformation: discrete and two scans

Implemented. The method is analyzable, but yet a bit tricky. It resulted wide slower than the others (optimizable though). The final result is not that impresive. Manhattan works better.

4 Skeletonization

Nice introduction about why is skeelton used can be found at the introduction of the paper **A one pass thinning algorithm and its parallel implementation**

4.1 A skeletonization algorithm bby maxima tracking euclidean dt

nice definition of distance transform on page 332.

On two scan distance transform:

Another algorithm only needs two operations in opposite scans of the picture: one scan is in the left-to-right top-to-bottom direction, and the other is in the right-to- left bottom-to-top direction. 116"t7) This does not require iterations and is efficient on conventional com- puters. Other algorithms apply the morphological erosions.t 18-20)

Why Euclidean over Chessboard and City-block: The city-block or chessboard distance measures are sensitive to the rotations of an object, but the Euclidean distance measure is rotation invariant. However, its square root operation is costly and the global nature of distance transformation is difficult to decompose into small neighborhood operations because of the nonlinearity of Euclidean distance variations

Good algorithm Very followable algorithm. Analyze well each stage to understand why is it taken that way.

In the paper: **Using ssm** in the introduction and other parts they introduce well why is it the medial axis descriptor good for representing skeleton.

4.1.1 Apex and base points

The apex and base points where computed following the algorithm presented in the paper
The base points are such who have at least 4 zeros around, this was changed to 5 because according to the worm shape there were many base points around the worm shape. This generated well the

real base (extreme) points of the worms with some issues: some real base points are missing (not much) and there are some points in places that do not correspond to an extreme of the worm. Need some refactoring here.

The apex points are such who are the local maxima in their 3x3 (8-n) neighborhood. These are generated well. The small width of the worms generates a 2-pixel width in some areas. This will be reduce with a pixel-reducer algorithm following the skeleton.

4.1.2 UpHill

The uphill generation looks for relative apex_base points to construct a connected skeleton. The connection is done following the *best possible path* which is assumed to be given by the higher value pixel in the three 45 degrees directional pixels.
This connected well many pixels approaching the shape well to a connected skeleton, but there are still missing pixels. The downhill processing should fix this.

Another problem regarding this is the apperance of *strange* or *ilogic* connections between wrongly generated pixels (normally base) and the partial skeleton.

4.1.3 Downhill and Uphill problems

Uphill gives the following problems:

- When a pixel is followed just by one and only one pixel it follows it in the second pixel direction. That doesn't allow to explore empty fields, just keeps follows the skeleton.
- When it has no pixel around then is marked, hoping that the downhill procedure can connect it with a partially built skeleton on UpHill.
- When is sorrounded by many then follows the best one with a directional neighbor. That is ok.

4.1.4 Thining (reduction)

The implemented reduction algorithm does not work properly. Disconnects some parts with the *elder* implementation of UpDownHill2. This last was changed, deleting the nonConnected check for the 1pixel intermediate neighbor. After this the obtained skeleton is thicker but totally connected.

- I am looking for literature to implement a best thining algorithm that works properly.
- When explaining take an eye to page 164 (anchor skeleton). They talk about thinning without touching special skeleton points, such as centres of maximal discs. NEEEH
- The need for thining skeleton comes for several things:
  - It makes easier and viable the segmentation of general image in sections. Otherwise it requires many considerations and it will be ambigous to define crossroads points

    - Once segmentated, the ”mixed” skeletons will be part of a unique worm representation. So it will be easy to calculate the optimization function cause it will depend on only one point each time.

- There is an algorithm in **Fast parallel algorithm for thining**. It seems to require to many iterations and calculations that are not explicitly mentioned, also is kinda dummy. Though, the obtained skeleton seems perfect, no missing points and totally connected.

4.1.5 Refine base points

After UpDownHill2 with TwoStepConnections gives us a totally connected skeleton we can refine the previously calculated base points.
A base point have to meet every of the following conditons

- It has to have at least one skeleton neighbor. (due to the total connection)
- If it has more than one neighbor and one of those is a base point// **This is not taked into consideration**

4.1.6 A fast algorithm for thinning digital patterns

Implementing the algorithm.

Description

It consists of two subiterations: one aimed at deleting the south-east boundary points and the north-west corner points while the other one is aimed at deleting the north-west boundary points and the south-east corner points. End points and pixel connectivity are preserved. Each pattern is thinned down to a ”skeleton” of unitary thickness.

Procedure

The algorithm removes the border or contour pixel that do not belong to a skeleton representation following the conditions that will be explained below.
A distance transformation of the image is used to find the current contour pixel for each run of the algorithm. A distance counter is started at 1 (border pixel) an augmented in each iteration, so pixels analized per iteration are the one who correspond to the distance counter in the image transformation (the contour pixels).

The conditions checked in each iteration are:

- First iteration
    - $2 \leq B(p1) \leq 6$ where $B(p)$ is the number of shape-pixels neighbors
    - $A(p1) = 1$ The number of consecutive 0-1, compared clockwise among the circular neighbors.
    - $p2 * p4 * p6 = 0$

    - $p4 * p6 * p8 = 0$

- Second iteration
- First two above
- $p2 * p4 * p8 = 0$
- $p2 * p6 * p8 = 0$

4.1.7 Detection of base-points

To detect the baes points the following conditions were taking into account

- The pixel has only one neighbor. In a connected 1-width skeleton any shape point with only one neighbor is inmediately a base point.
- The 1-pixel width reduction method does not consider diagonal connections, so just a 4-neighbors connections is taken into account. This makes that when created a diagonal a manhattan two step is done. So the following condition has to be accomplished for having a base point: There is one and just one *group* directional neighbor. The groups are conformed by consecutive neighbors taken them clockwise.

Works perfectly and in a real-worm shape the points detected as base points are just the right ones. If there is still some noise, these will be recognized as base points when corresponding.