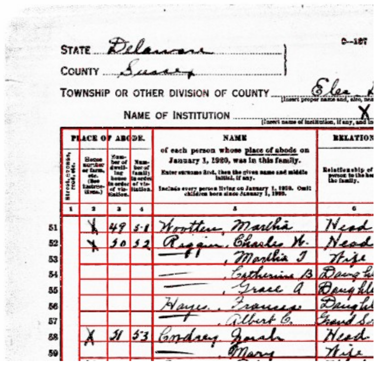
Intelligent Pen: A Least-Cost Search for Tracing of Handwriting

Kevin Larry Bauer
Brigham Young University
March 2014

Introduction

A subject of increasing interest in document processing and analysis is the movement to index the handwriting contained in scanned images of documents. Many of these documents are historical in nature such as census records, birth and death records, parish and church records, journals, marriage certificates, and lists of passengers on ships.



STATE Delaware

COUNTY Sussex

TOWNSHIP OR OTHER DIVISION OF COUNTY Elea

[Insert proper name and, also, nam

NAME OF INSTITUTION X

[Insert name of institution, if any, and in

	PLACE OF ABODE.				NAME	RELATION
	Street, avenue, road, etc.	House number or farm, etc. (See instructions.)	Number of dwelling house in order of visitation.	Number of family in order of visitation.	of each person whose place of abode on January 1, 1920, was in this family. Enter surname first, then the given name and middle initial, if any. Include every person living on January 1, 1920. Omit children born since January 1, 1920.	Relationship of person to the head of the family.
	1	2	3	4	5	6
51		X	49	51	Wootten, Martha	Head
52		X	50	52	Riggin, Charles H.	Head
53					——— , Martha J	Wife
54					——— , Catherine B	Daughter
55					——— , Grace A	Daughter
56					Hayes, Frances	Daughter
57					——— , Albert C.	Grand Son
58		X	51	53	Cordrey, Josiah	Head
59					——— , Mary	Wife

Figure 1a. Detail of a 1920 census form with grid lines highlighted

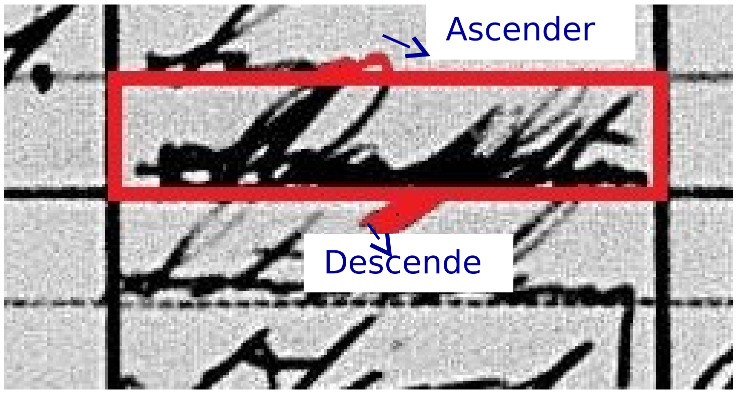


Figure 1b. Example of a word with an ascender and a descender

Before such records can be indexed, either manually or through automatic handwriting recognition, the handwriting must first be extracted. In seeking to extract the strokes of the handwriting in historical documents the most common problem faced is the descender problem, where the text goes below the printed line (Figure 1b). This is usually caused by letters with a “tail” such as ‘y’, ‘g’, ‘p’, and ‘q’. In other cases the text may include ascenders that go above the form line (usually for capital letters or letters like ‘l’, ‘d’, and ‘b’). In each of these cases an intelligent method for detecting and extracting the entire handwritten word is vital for effective handwriting recognition to take place.

This paper presents a novel approach to extracting the strokes of handwritten text in historical documents utilizing a least-cost path search algorithm. By tuning the parameters of the cost function the algorithm can be made resilient to missing line segments (gaps), extra line segments, interference from the form lines and other noise.

Related Work

Typically the goal of stroke extraction algorithms is to be able to apply on-line handwriting recognition methods to static, offline handwriting images. This is usually done by using a medial axis transform or some other thinning algorithm to extract a skeleton: a pixel-wide representation that outlines the basic shape of the handwriting.Various techniques exist for extracting such skeletons, such as diffusion maps [5] or line thinning [7, 8]. Figure 2 gives an example of a skeleton obtained by line thinning.

Once the skeleton is obtained, some tracing method is used to find an ordered path through the skeleton, either a local line-following algorithm or a global graph search. The key assumption made by algorithms such as these that rely on skeletonization is that the skeleton can be successfully extracted without loss of data or the introduction of noise, and the publications contain sparse information on what validation has been performed to this effect. To overcome these challenges a new approach is needed that is more robust to noise and works with older, degraded handwriting, and is tolerant to the existence of form lines that overlap the handwriting. Rather than create a skeleton or graph representation of the image, we propose an algorithm that would operate directly on the grayscale pixels of the original image.

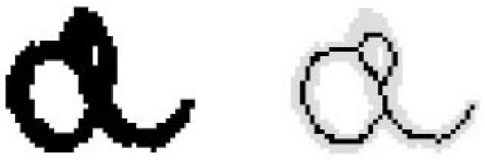


Figure 2. An example of a handwritten lower-case ‘a’ and its skeleton, obtained by line thinning. Taken from a figure in the paper by L’Homer [8].

Figure 3 gives some examples of the kinds of challenges this algorithm will have to overcome, such as gaps (Figure 3a), extra line segments introduced by noise (Figure 3b), and interference between adjacent cells (Figure 3c).

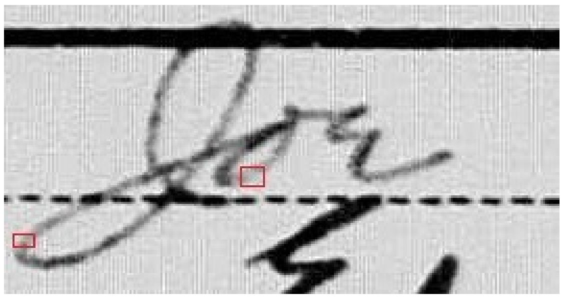


Figure 3a. Handwritten words in scanned documents are often degraded, with low contrast and gaps in the strokes, as highlighted by the red rectangles.

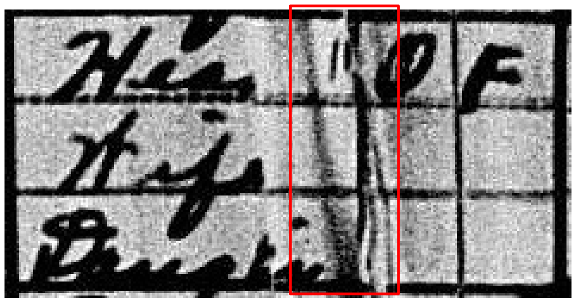


Figure 3b. Damage to the original document requires that the system be robust to noise and extraneous lines.

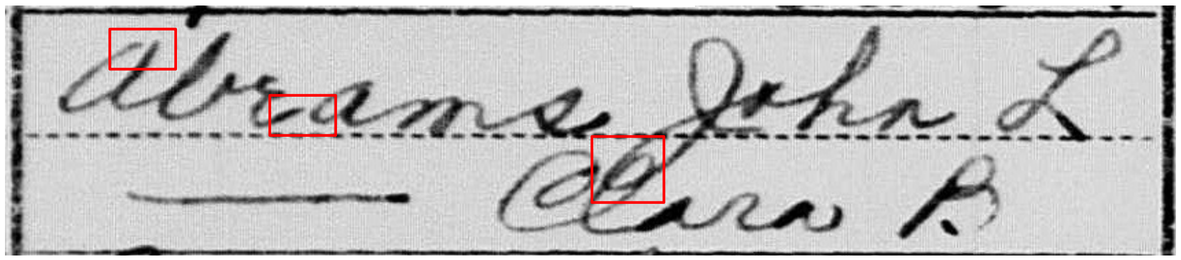


Figure 3c. Even in a low noise environment the handwriting may contain gaps or other interference.

Methods

To accomplish these goals, we will use a dataset of scanned document images from the 1920 and 1940 US census. An example is seen in Figure 1a.

Our algorithm, which we will call Intelligent Pen, will use its search function to find a path from one end of a handwritten word to the other. An optimal path in this context is one that goes through every stroke of the word, following the stroke order of the document’s author, and ignoring any gaps or extraneous strokes. The path obtained by our algorithm will be piecewise-optimal in the sense that the final path will be created by taking the union of several locally optimal path segments. Although such a path will not guarantee global optimality, intuitively it should provide a good approximation so long as it follows the handwriting from start to finish.

Finding Starting Points

The algorithm begins by finding a set of potential starting points (the red dots in Figure 4). This can be done by creating a grid as seen in Figure 4. The width and height of the grid could be determined by an estimate of average stroke thickness, or adjusted to make the search more or less fine-grained. The neighboring pixels of each vertex in the grid will be examined, and the vertices that lie on a clearly defined stroke will be chosen as potential starting points. This will be determined by examining the neighborhood around the pixel for pixel intensity and gradient direction.

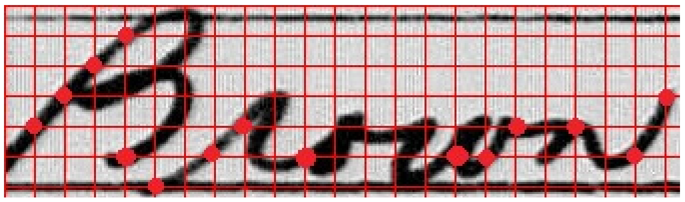


Figure 4. Using a grid search to find potential starting points

The reason for doing a grid search to identify starting points is that the boundaries of the form data are not known beforehand. The actual "pen-down" point could be difficult or impossible to identify without this prior knowledge, and it may lie outside the form cell itself. However, the stroke extraction method described below compensates by performing a multi-directional search that allows it to function properly regardless of what coordinates are chosen as the initial starting points.

Cost function

As stated above, the algorithm will employ a least-cost path search to extract strokes from the image. Table 2 and Equation 2 summarize the cost function and what kinds of features it will include.

Image Feature	Rationale	Formulation
Pixel value	Give preference to darker pixels	f_i
Change in direction	Avoid jumping to different strokes at intersection	f_d
Change in line thickness	Avoid jumping to nearby strokes	f_t
Is the pixel on a form line?	Avoid extracting form lines as part of a stroke	f_l
Distance to nearest gridline (if outside cell bounds)	Avoid straying too far outside the cell	f_O
Has this pixel been visited?	Avoid retracing the same path	f_v

Table 2

$$C(p) = w_i f_i + w_d f_d + w_t f_t + w_l f_l + w_O f_O + w_v f_v \qquad [2]$$

Other features discovered during the testing process will also be added into the equation as needed. The *w* terms are weights which will be tuned using a training set of handwriting images that have been traced over by hand to manually extract the strokes. The weights chosen will be those with the highest accuracy in finding strokes and stroke order.

The problem of tracing a piece of handwriting in an image can therefore be framed as the minimum cost path through a graph where each pixel is a node and the edge weight is the result of applying Equation 2. Appendix A gives a more detailed overview of the least-cost path search, which is based on Dijkstra's algorithm for finding the shortest path through a graph.

Stroke Extraction

The goal of this process is stroke extraction, where a stroke is defined as a piece of a handwritten word. Several strokes, when merged together, form what will be referred to as a "trace-line". A "trace-line" is here defined as an ordered sequence of pixels, containing the union of all pixels from the ink added to the page between the author placing the pen on the page and lifting it up again.

To achieve this, all start points found by the grid search in Figure 5 are placed on the stack, $\boldsymbol{S}$. The top start point is then popped off the stack and the stroke extraction process proceeds as shown in Figure 5. Beginning at a start point $\boldsymbol{s}$, the algorithm will create a circle of radius r centered at $\boldsymbol{s}$, and choose several terminal points t_i evenly spaced around the circle. (Figure 5a). For each t_i, the least-cost path from $\boldsymbol{s}$ to t_i will be found (Figure 5b). As paths are found for each t_i, some of these paths will have sections that overlap or have consensus with one or more other paths (Figure 5c). This is because the search is optimal, meaning that the best path from s to C will be the same regardless of whether the end goal is to arrive at t_i or at t_j.

Algorithm 1: Consensus Stroke Extraction

```
INPUTS:
  A stack, $\boldsymbol{S}$, of potential starting points
  A Global set of consensus strokes $\boldsymbol{G}$
OUTPUTS:
  A consensus stroke $\boldsymbol{C}$
  A set of additional potential starting points that are added to the stack

while $\boldsymbol{S}$ is not empty:
  get a starting point, $\boldsymbol{s}$, from $\boldsymbol{S}$
  if $\boldsymbol{s}$ is already on or near a point in $\boldsymbol{G}$:
    remove $\boldsymbol{s}$ from $\boldsymbol{S}$ and begin again
  generate a set of free points, $\boldsymbol{t_i}$, using a circle of radius $\boldsymbol{r}$ (Figure 5a)
    initialize a local consensus path $\boldsymbol{C}$ (as empty set of pixels)
    for each free point $\boldsymbol{t_i}$,:
      Path $\boldsymbol{p_i}$ = FindBestPath($\boldsymbol{s}$,$\boldsymbol{t_i}$) // FindBestPath() = Dijkstra's search //

  for each pair of paths $\boldsymbol{p_1}$, $\boldsymbol{p_2}$:        // $\boldsymbol{p_1}$ is the path from $\boldsymbol{s}$ to $\boldsymbol{t_i}$, $\boldsymbol{p_2}$ is the path from $\boldsymbol{s}$ to $\boldsymbol{t_j}$. (Figure 5c)
    for each pixel $\boldsymbol{q_1}$ in $\boldsymbol{p_1}$:
      for each pixel $\boldsymbol{q_2}$ in $\boldsymbol{p_2}$:
        if $\boldsymbol{q_1}$ = $\boldsymbol{q_2}$ and $\boldsymbol{q_1}$ is not in any path in $\boldsymbol{G}$: // Because the search is optimal shared points will be
                                                          // part of the globally optimal path.
          Add $\boldsymbol{q_1}$ and $\boldsymbol{q_2}$ to $\boldsymbol{C}$
      Add each endpoint, $\boldsymbol{e_i}$ of $\boldsymbol{C}$ to $\boldsymbol{S}$ (Figure 5d)
      Add $\boldsymbol{C}$ to $\boldsymbol{G}$
```

Each of these overlapping path sections, which we will refer to as consensus strokes, will be saved. Adjacent consensus strokes will be merged to form trace-lines (Figure 5i and Algorithm 2). In Figure 5d we see that the example start point is connected to 4 consensus paths terminating at points A, B, C, and

D. Each of these 4 points is then placed on the stack with the other potential start points. The top one is then popped off the stack and the algorithm begins again with A as the new start point, recursively exploring each start point until the stack ***S*** is empty. Once ***S*** is empty, the algorithm terminates. The pseudocode for this process is shown above in Algorithm 1.

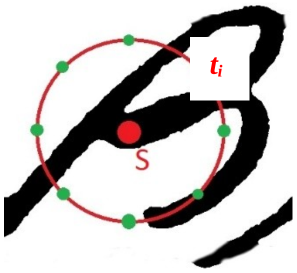


Figure 5a. To begin, evenly-spaced free points, t_i are chosen on a circle of radius ***r*** centered at the start point ***s.***

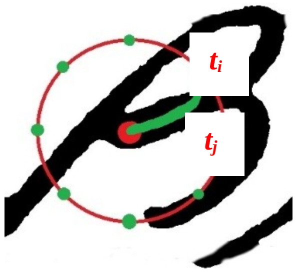


Figure 5b. The least-cost path to each of these free points (t_i and t_j) is calculated.

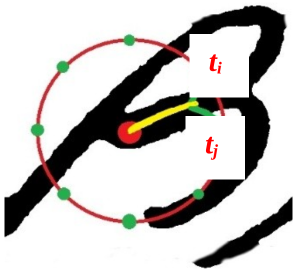


Figure 5c. Pixels that lie on or near the path to more than one free point are saved as consensus strokes.

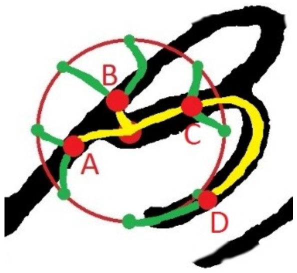


Figure 5d. The endpoints of these consensus strokes become new start points, A, B, C and D.

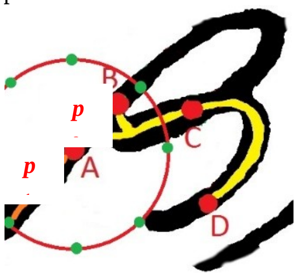
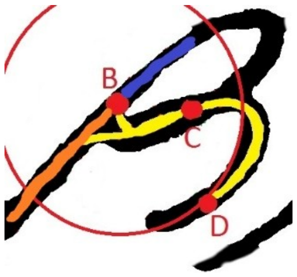


Figure 5e. Using A as a startpoint, new consensus strokes, p_1 and p_2 (orange) are found.



Figrue 5f. Using point B as a startpoint the blue consensus stroke is found.

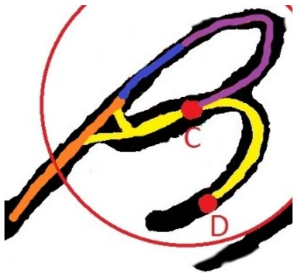


Figure 5g. Likewise, the purple stroke is found by starting at point C.

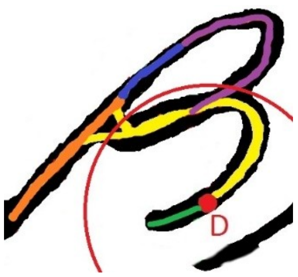


Figure 5h. Finally D is popped off the stack and the green consensus stroke is obtained.

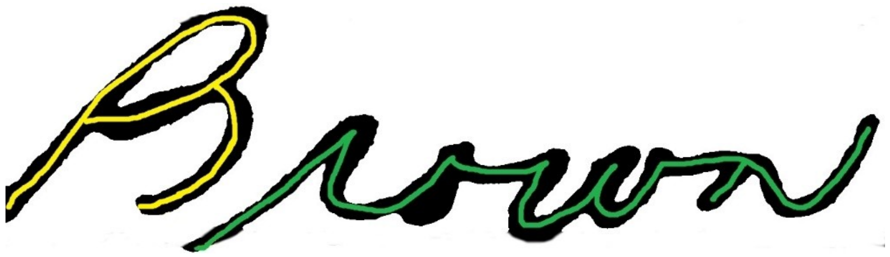


Figure 5i. Once consensus paths are found for D the algorithm continues with whatever was on the stack below ***S***. Consensus paths whose starting points are the endpoints of another path (such as the paths from A, B, C, D, and ***S***) are combined into a single stroke (the yellow path). Consensus paths whose start points are not the end point of another consensus path will be treated as a separate stroke.

Combining Strokes into Trace-Lines:

Each iteration of the above process will extract a consensus path composed of pixels that lie on multiple shortest paths from a start point to an endpoint. Once this process has terminated individual consensus paths will be combined to form strokes. This will be fairly simple since the endpoints of each consensus path are used as starting points for the next iteration. The pseudocode for this step of the process looks like this:

Algorithm 2: Stroke Merging

```
INPUTS:
   A set of Consensus strokes $G$
OUTPUTS:
   A set of trace-lines, ▯

for each path $p_1$ in $G$:
   for each path $p_2$ in $G$:
      if the start point of $p_2$ is the start point of $p_1$:
         reverse the order of $p_2$ so its start point is now its end point
      if start point of $p_2$ is endpoint of $p_1$
      or start point of $p_1$ is endpoint of $p_2$
          $p_1 = p_1$ ▯ $p_2$
          Remove $p_2$ from $G$
```

After one iteration, p_1 will be a complete trace-line. Note that by this definition a trace-line may branch, such as at point A in Figure 5. Because they are not connected to p_1, any remaining paths in G must necessarily lie on different trace-lines (for example, the yellow and green lines in Figure 5i), which will be identified on subsequent iterations. The trace-lines are then ordered, with the longest path being first (since this is probably the main skeleton of the word), followed by all the shorter paths in a left to right order (since these are probably things like crossing 'T's or dotting 'I's).

As mentioned above, because of the Bellman Principle of Optimality, any subsection of an optimal path is itself an optimal path, which means each consensus path is an optimal path between its endpoints. Although the converse is not necessarily true (the union of multiple adjacent optimal paths is not guaranteed to be an optimal path), by carefully choosing the radius r and the parameters of the cost function the algorithm should follow the handwriting well enough to output trace-lines that are close to optimal for the entire word.

Bounds Extraction

As the algorithm proceeds it keeps track of each point that lies on a stroke. The boundaries of the word are therefore given by drawing a rectangle around the min and max x and y coordinates of the union of the extracted strokes.

Validation

One challenge in validating the accuracy of trace-lines is that little work has been done in extracting complete trace lines from offline historical documents. Also, validation methods vary between different proposed methods, with different aspects of the handwriting being considered as ground truth. In seeking to validate the effectiveness of our method we need to look at its ability to solve the ascender/descender problem as well as its ability to extract strokes into accurate trace lines. We propose two forms of validation for our method. The first will show quantitatively how reasonably the trace-lines extracted by Intelligent Pen follow the handwriting, while the second will show qualitatively how Intelligent Pen overcomes the limitations of skeletonization algorithms in the context of historical document analysis.

Kennard et al. [36] describe a handwriting recognition algorithm that achieves over 90% accuracy for in-vocabulary words. Kennard's algorithm accomplishes this through the use of a medial axis transform and image morphing to align two handwritten images. Since we have access to a working implementation of this method, this presents an ideal opportunity for validating the accuracy of our extracted trace-lines. We will use a dataset composed of “clean” handwriting images (Figure 6a). “Clean” in this case meaning free from the noise and degradation typically observed in historical documents. We will obtain a measure of how well Intelligent Pen’s trace-lines actually model the handwriting by comparing our trace-lines against the medial axis transform. The intuition behind this is that for such “clean” images the medial axis should represent a statistically ideal representation of the handwriting. If our trace-lines compare closely with the medial axes, this will allow us to infer that they could also be used to achieve comparable accuracy in handwriting recognition.

To perform this validation we will treat our trace-lines as binary images and compare them with Kennard’s medial axes using distance maps, where points on the medial axis will have distance 0 and points emanating out from the medial axis will have positive signed distances (Figure 6c). Therefore, by overlaying a trace-line on the corresponding distance map we can obtain a similarity measure between the trace-line and the medial axis transform by summing the distance values of each.

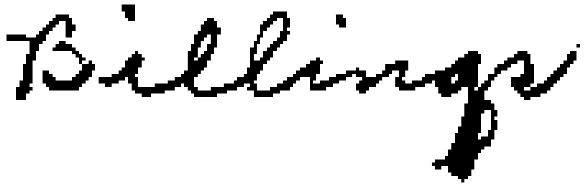


Figure 6a. An example from [37] of a “clean” piece of handwriting.

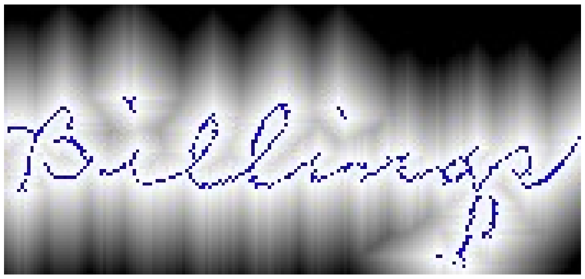
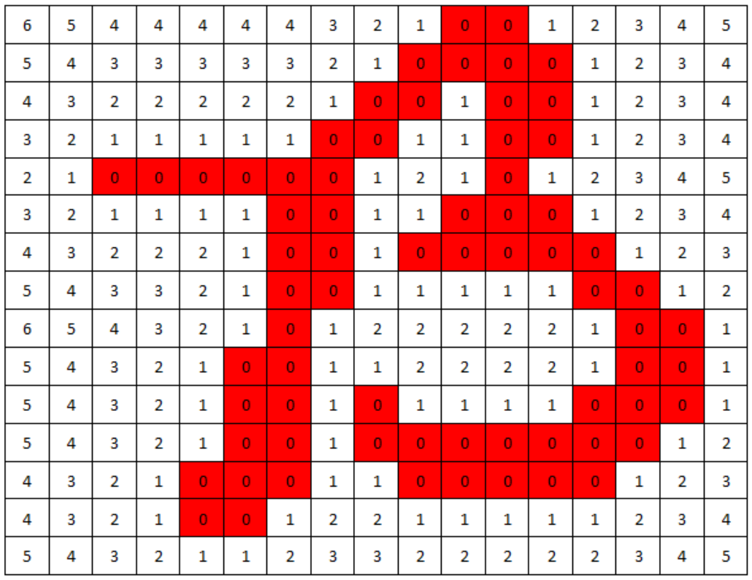


Figure 6b. The distance map associated with Figure 6a, also taken from [37]. Darker shades represent greater distance from the medial axis. The medial axis is outlined in blue.



6	5	4	4	4	4	4	3	2	1	0	0	1	2	3	4	5
5	4	3	3	3	3	3	2	1	0	0	0	0	1	2	3	4
4	3	2	2	2	2	2	1	0	0	1	0	0	1	2	3	4
3	2	1	1	1	1	1	0	0	1	1	0	0	1	2	3	4
2	1	0	0	0	0	0	0	1	2	1	0	1	2	3	4	5
3	2	1	1	1	1	0	0	1	1	0	0	0	1	2	3	4
4	3	2	2	2	1	0	0	1	0	0	0	0	0	1	2	3
5	4	3	3	2	1	0	0	1	1	1	1	1	0	0	1	2
6	5	4	3	2	1	0	1	2	2	2	2	2	1	0	0	1
5	4	3	2	1	0	0	1	1	2	2	2	2	1	0	0	1
5	4	3	2	1	0	0	1	0	1	1	1	1	0	0	0	1
5	4	3	2	1	0	0	1	0	0	0	0	0	0	0	1	2
4	3	2	1	0	0	0	1	1	0	0	0	0	0	1	2	3
4	3	2	1	0	0	1	2	2	1	1	1	1	1	2	3	4
5	4	3	2	1	1	2	3	3	2	2	2	2	2	3	4	5

Figure 6c. Detail of the distance map for the letter B from Figure 6a. The medial axis (marked in red), has distance zero. Other pixels are assigned a value based on their 4-distance to the nearest medial axis pixel.

Where a coordinate on the trace-line lands on a distance of 0, there is a point on the medial axis representation that corresponds exactly to the trace-line. Therefore, summing over all distances corresponding to a given trace-line will tell us how closely our trace-line corresponds to the medial axis.

Lower sums imply stronger similarity; higher sums reveal greater differences. The summed distance will represent the total distance between the trace-line and the medial axis. Dividing this sum by the number of points in the medial axis will give us the average, per-pixel distance.

We can then make use of a chi-squared test to report the similarity of our trace-lines with the medial axes across a set of 1000 words, using the medial axis as the expected value and the trace-line coordinates as the observed value. This will capture the squared difference between the two representations, amplifying places of greater disparity. A low chi-square score will confirm the validity of our extracted trace-lines. In other words Intelligent Pen method can be considered valid if it consistently obtains trace-lines with a low chi-square score.

Having shown that Intelligent Pen obtains a good approximation of handwriting in “clean” situations, the next step will be to show that it can perform better than existing methods in less ideal circumstances. Much of the existing work is based on skeletonization algorithms which are known to have problems with handwriting such as that seen in older, noisy historical documents. Therefore, in our comparisons we will also isolate words or word components that have breaks, drop-outs, noise, etc. that are known to be problematic with the skeletonization techniques and demonstrate how the optimal trace-line hops across these breaks. We will also show how our trace-line follows faint strokes that are not preserved with binarization and subsequent skeletonization. Finally, we will demonstrate the robustness of our trace-lines in areas of handwriting that otherwise produce noise in the medial axis skeleton.

Our trace-lines will attempt to preserve stroke order and we believe stroke order is an important feature for subsequent handwriting recognition. However, for purposes of comparison, and because we do not have stroke order information for the medial axis and skeletonization techniques, we have chosen to set stroke order aside and simply treat the trace-line as a binary image.

Initial Results:

A complete working version of the algorithm is still in development. Thus far a simple working version of the least-cost search function has been created using only the f_i and f_d coefficients. The initial results, shown in Figures 7 and 8, are quite promising.

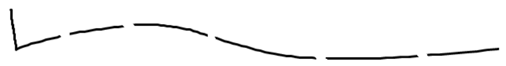


Figure 7a. One of the main challenges faced by existing methods is how to handle gaps in the handwriting.

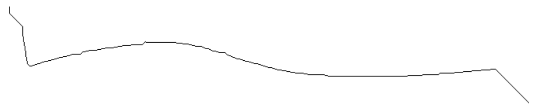


Figure 7b. The least-cost path obtained by searching from the top-left corner to the bottom-right corner of 7a. Only the f_i and f_d coefficients were used in the cost function.

Figure 7a shows a simple example that poses a challenge for existing methods: a smooth line with multiple gaps. We can see from Figure 7b that Intelligent Pen's search function is able to find a smooth path that passes through the line and correctly jumps across the gaps.

In Figure 8 we see a more complicated example. Instead of jumps the stroke doubles back several times and there are several stray strokes simulating noise. Intelligent Pen is able to ignore the noise and correctly follow the darker line, with the result shown in Figure 8b.

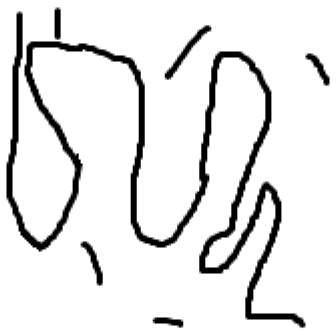


Figure 8a. A stroke with some added noise and curves.

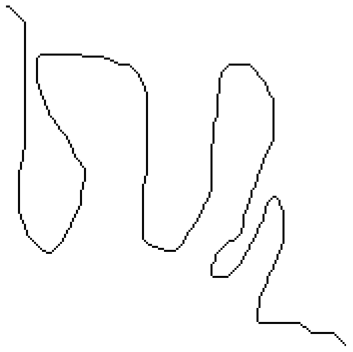


Figure 8b. The least-cost path obtained for 8a using the same parameters as 7b. Note that the path ignores the noise and follows the smoothest path through the image.

Conclusion:

We have shown how a least-cost search function can be used to improve on existing stroke extraction methods while simultaneously solving the ascender/descender problem. While further work is needed to refine the algorithm and its parameters, the initial results are promising and able to overcome some common pitfalls, and by framing the problem as a least-cost search the algorithm can be easily optimized for specific needs and situations.

Bibliography

1. http://en.wikipedia.org/wiki/FamilySearch
2. http://www.mormonnewsroom.org/article/fast-facts-about-familysearch-indexing
3. Clawson, et al. "Automated Recognition and Extraction of Tabular Fields for the Indexing of Census Records," *Document Recognition and Retrieval (DRR) 2013*. Feb 2013
4. Gehring, Jake. Personal Correspondence with the author.
5. Daher, H.; Gaceb, D.; Eglin, V.; Bres, S.; Vincent, N., "Ancient Handwritings Decomposition Into Graphemes and Codebook Generation Based on Graph Coloring," *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, pp.119,124, 16-18 Nov. 2010
6. Tan, J.; Lai, J.; Zheng, W.; Suen, C.Y., "A Novel Approach for Stroke Extraction of Off-Line Chinese Handwritten Characters Based on Optimum Paths," *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*, pp.786,790, 18-20 Sept. 2012
7. Huang, T.; Yasuhara, M., "A total stroke SLALOM method for searching for the optimal drawing order of off-line handwriting," *Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on*, vol.3, pp.2789,2794 vol.3, 22-25 Oct 1995
8. L'Homer, E., "Extraction of strokes in handwritten characters," *Pattern Recognition*, Volume 33, Issue 7, pp. 1147-1160 July 2000
9. Mortensen, E. N., & Barrett, W. A. (1998). "Interactive segmentation with intelligent scissors," *Graphical models and image processing*, Volume 60, Issue 5, pp. 349-384.9
10. Everton B. Lacerda, Carlos A.B. Mello, "Segmentation of connected handwritten digits using Self-Organizing Maps," *Expert Systems with Applications*, Volume 40, Issue 15, Pages 5867-5877 1 Nov 2013
11. Y. Liang, M.C. Fairhurst, R.M. Guest, "A synthesized word approach to word retrieval in handwritten documents," *Pattern Recognition*, Volume 45, Issue 12, December 2012, Pages 4225-4236
12. Mortensen, E.; Morse, B.; Barrett, W.; Udupa, J., "Adaptive boundary detection using `live-wire' two-dimensional dynamic programming," *Computers in Cardiology 1992, Proceedings of* , vol., no., pp.635,638, 11-14 Oct 1992
13. Elbaati, A.; Boubaker, H.; Kherallah, M.; Ennaji, A.; Ennaji, A.; Alimi, A.M., "Arabic Handwriting Recognition Using Restored Stroke Chronology," *Document Analysis and Recognition, 2009. ICDAR '09. 10th International Conference on* , vol., no., pp.411,415, 26-29 July 2009
14. Tselios, K.; Zois, E.N.; Nassiopoulos, A.; Economou, G., "Fusion of directional transitional features for off-line signature verification," *Biometrics (IJCB), 2011 International Joint Conference on* , vol., no., pp.1,6, 11-13 Oct. 2011
15. Cheng-Lin Liu, In-Jung Kim, Jin H. Kim, "Model-based stroke extraction and matching for handwritten Chinese character recognition," *Pattern Recognition*, Volume 34, Issue 12, December 2001, Pages 2339-2352
16. Lin, Feng; Xiaoou Tang, "Off-line handwritten Chinese character stroke extraction," *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol.3, no., pp.249,252 vol.3, 2002
17. Xiufen Ye; Weiping Hou; Weixing Feng, "Off-line handwritten signature verification with inflections feature," *Mechatronics and Automation, 2005 IEEE International Conference*, vol.2, no., pp.787,792 Vol. 2, 29 July-1 Aug. 2005
18. Claudio De Stefano, Angelo Marcelli, Antonio Parziale, Rosa Senatore, "Reading Cursive Handwriting," *Frontiers in Handwriting Recognition, International Conference on*, pp. 95-100, 2010 12th International Conference on Frontiers in Handwriting Recognition, 2010

19. Chen Yan; Leedham, G., "Decompose-threshold approach to handwriting extraction in degraded historical document images," *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pp.239,244, 26-29 Oct. 2004

20. Clawson, R. and Barrett, W., “Extraction of Handwriting in Tabular Document Images,” *Family History Technology Workshop at Rootstech*, 2012.

21. Pavlidis, Theo., “A vectorizer and feature extractor for document recognition”, *Computer Vision, Graphics, and Image Processing*, Volume 35, Issue 1, July 1986, Pages 111-127

22. Simon, J. -C, "Off-line cursive word recognition," *Proceedings of the IEEE* , vol.80, no.7, pp.1150,1161, Jul 1992

23. Yu Qiao; Yasuhara, M., "Recovering dynamic information from static handwritten images," *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on* , vol., no., pp.118,123, 26-29 Oct. 2004

24. Yu Qiao; Makoto Yasuhara, "Recovering Drawing Order from Offline Handwritten Image Using Direction Context and Optimal Euler Path," *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* , vol.2, no., pp.II,II, 14-19 May 2006

25. Yu Qiao; Makoto Yasuhara, "Recover Writing Trajectory from Multiple Stroked Image Using Bidirectional Dynamic Search," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on* , vol.2, no., pp.970,973, 0-0 0

26. Emli-Mari Nel, J.A. du Preez, B.M. Herbst, Verification of dynamic curves extracted from static handwritten scripts, *Pattern Recognition*, Volume 41, Issue 12, December 2008, Pages 3773-3785

27. Lee, S.; Pan, J.C., "Offline tracing and representation of signatures," *Systems, Man and Cybernetics, IEEE Transactions on* , vol.22, no.4, pp.755,771, Jul/Aug 1992

28. G. Boccignone, A. Chianese, L.P. Cordella, A. Marcelli, Recovering dynamic information from static handwriting, *Pattern Recognition*, Volume 26, Issue 3, March 1993, Pages 409-418

29. Christian Viard-Gaudin, Pierre-Michel Lallican, Stefan Knerr, Recognition-directed recovering of temporal information from handwriting images, *Pattern Recognition Letters*, Volume 26, Issue 16, December 2005, Pages 2537-2548

30. Lallican, P.-M.; Viard-Gaudin, C., "A Kalman approach for stroke order recovering from off-line handwriting," *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on* , vol.2, no., pp.519,522 vol.2, 18-20 Aug 1997

31. Allen, C.R.; Navarro, A., "The recognition of Roman hand-written characters using dynamic information recovery," *Image Processing and Its Applications, 1997., Sixth International Conference on* , vol.2, no., pp.741,745 vol.2, 14-17 Jul 1997

32. Doermann, D.S.; Rosenfeld, Azriel, "Recovery of temporal information from static images of handwriting," *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference on* , vol., no., pp.162,168, 15-18 Jun 1992

33. Hirobumi Nishida, An approach to integration of off-line and on-line recognition of handwriting, Pattern Recognition Letters, Volume 16, Issue 11, November 1995, Pages 1213-1219

34. Nagoya, T.; Fujioka, H., "Recovering drawing order from static handwritten images using probabilistic tabu search," *TENCON 2011 - 2011 IEEE Region 10 Conference* , vol., no., pp.379,384, 21-24 Nov. 2011

35. http://en.wikipedia.org/wiki/Bellman_equation#Bellman.27s_Principle_of_Optimality

36. Kennard, D.J.; Barrett, W.A.; Sederberg, T.W., "Word Warping for Offline Handwriting Recognition," *Document Analysis and Recognition (ICDAR), 2011 International Conference on* , vol., no., pp.1349,1353, 18-21 Sept. 2011

37. Kennard, D. J. “Warping-Based Approach to Offline Handwriting Recognition.” PhD Dissertation, Brigham Young University, Apr. 2013

Appendix A: Least-Cost Search Algorithm

102	113	243	213	201	103	54	59	50	10
212	236	226	206	103	82	102	132	212	252
248	240	208	112	53	144	223	234	219	213
232	241	101	75	138	210	255	244	219	208
205	142	91	120	218	223	212	248	207	254
202	112	134	250	251	205	216	240	232	230

Figure A1. The initial cost matrix for a simple image with the start point marked in green and the destination point marked in red.

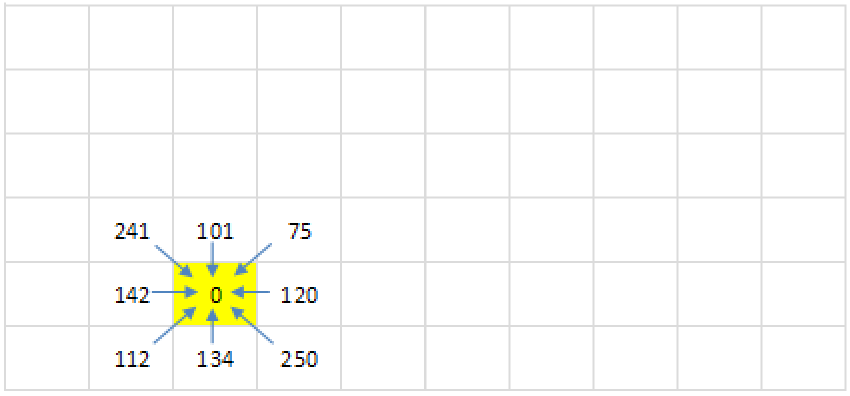


Figure A2. The start point is initialized with a cumulative cost of zero and marked as visited (throughout this figure the visited nodes are marked in yellow). All of its neighbors are also added to the graph. Their cumulative cost is simply their value from the initial cost matrix.

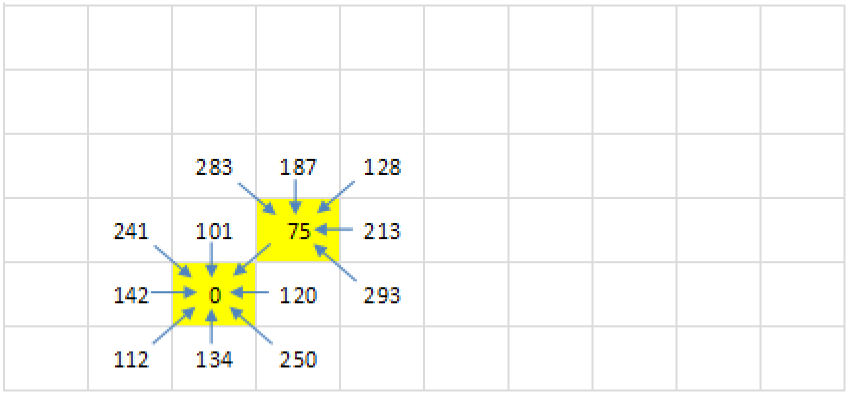


Figure A3. The neighbor with the lowest cost is added and marked as visited. Each of its neighbors is added with a cumulative cost of their own value from the initial cost matrix, plus that of the current node (75 in this case).

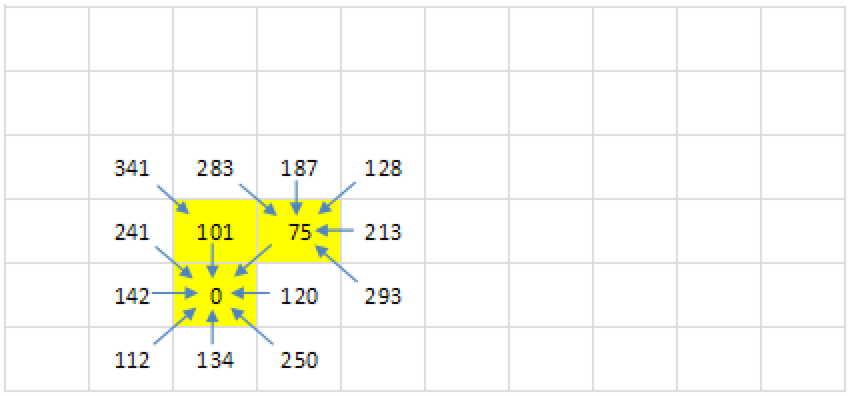


Figure A4. The node with the next lowest cost (101) is visited and all its neighbors that aren't already in the graph are added with a cumulative cost of 101 plus their initial cost.

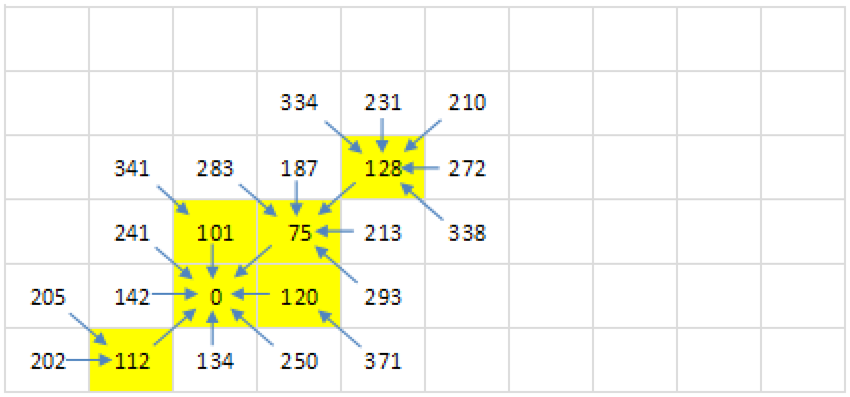


Figure A5. The process continues, sometimes moving in the direction of the target point, but not always.

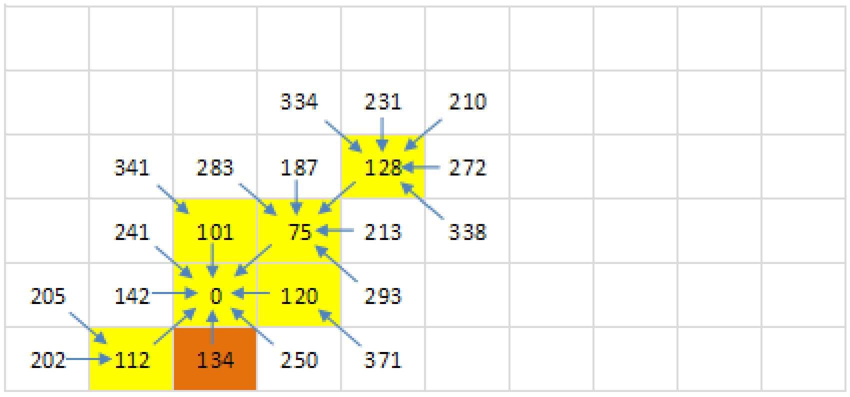


Figure A6. At this point the algorithm visits a node with no neighbors not already in the graph. This constitutes a dead-end, and the node can be discarded since it will not be part of the optimal path to the target node. Here and throughout the diagram such dead-end nodes will be marked in orange.

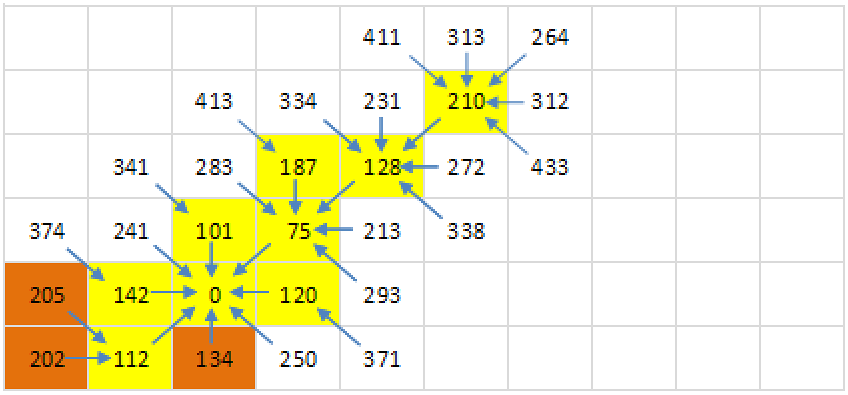


Figure A7. Here the graph is shown after 11 time-steps.

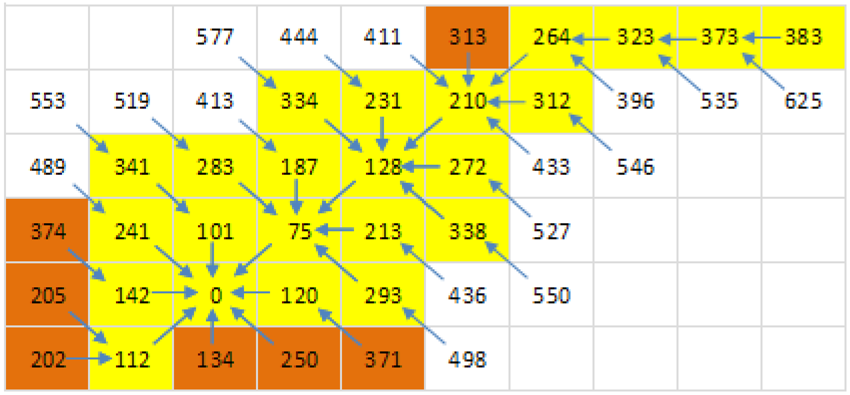


Figure A9: After 29 time-steps the search algorithm has reached the target node.

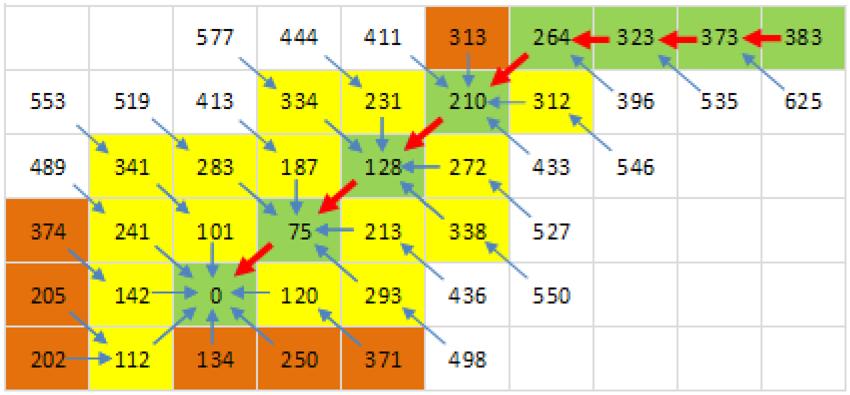


Figure A10: Once the target node is reached the optimal path is found by following the trail of pointers back to the start node.