STITCHING IMAGES BACK TOGETHER

FRITHJOF LUTSCHER, JENNY MCNULTY, JOY MORRIS, AND KAREN SEYFFARTH

1. Introduction

When a large visual is scanned into a computer in pieces, or printed out across multiple sheets of paper, distortions are often introduced in the scanning or printing process that make it impossible to fully reconstruct the original visual ideally. For clarity, we refer to the pieces of the visual as images, and the whole visual as the picture.

Typically, either scanning or printing is done with some overlap between adjacent images, that makes it possible to find very good relative locations for placing any pair of images that were adjacent. Unfortunately, the distortion often means that not all of the pairwise relative locations can be achieved simultaneously.

We present a graph theoretic approach to this problem. This approach is purely local in the sense that we only use the fact that any given pair of adjacent images can be stitched together perfectly, i.e., all features match, due to the offset data from pairwise correlation. At present, these local correlations are used for stitching following a certain pattern. We propose different patterns which should give better global results. To that end, we formulate the problem in a graph theoretic way and introduce some error measures. We determine lower bounds for the different measures of error. We find and compare several alternative stitching patterns which have smaller errors than the one currently used.

1.1. Notation.

We assume the images are arranged in a rectangular shape of m rows and n columns, where $m \geq n$. Define the graph G by letting each image correspond to a vertex. Two vertices are joined by an edge if the corresponding images are adjacent. Hence, G is an $m \times n$ grid graph. We denote by $E(G)$ the edges of G, and for adjacent $u, v \in G$ we write $uv \in E(G)$.

We use the stitching pattern to define a *spanning tree* T of G, i.e., a connected subgraph without cycles. An edge of G is an edge of T if the corresponding images are stitched together perfectly according to the offset values. The current method of row-wise stitching corresponds to the row

Joy Morris gratefully acknowledges support from NSERC grant # 40188.

pattern given by the solid lines in Figure 1. Here, the images are stitched together row-wise and then the strips are stitched together along a "spine."

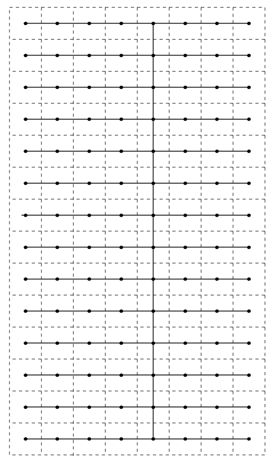

FIGURE 1. 14×8 **Row Pattern**, R

We assume that the error between u and v with respect to the true offset values increases with the distance between u and v in the tree T. To quantify this error, we define the following. For each $uv \in E(G)$,

$d'_T(u,v)$ denotes the distance between u and v in the tree T;

$d_T(u,v) = d'_T(u,v) - 1$;

$S(T) = \Sigma_{uv \in E(G)} d_T(u,v)$ (i.e., the sum of all the distances, in T, between adjacent vertices of G);

$M(T) = \max_{uv \in E(G)} d_T(u,v)$.

Note that we subtracted 1 from the distance in the tree to get d_T because we assume that pairwise stitching is free of error. We address the following optimization problems.

(1) Find a spanning tree T of G that minimizes $S(T)$, the sum of the distances of the tree.
(2) Find a spanning tree T of G that minimizes $M(T)$, the maximum of the distances of the tree.
(3) Find a spanning tree T of G that minimizes $S(T)$, the sum of the distances of the tree, subject to $M(T)$ being as small as possible.

It is worth noting here that the essential concept here of attempting to somehow preserve distance from a graph to a spanning tree, has been studied extensively (see, for example, [1] or [2]). A *tree t-spanner* of a graph G is a spanning tree in which the distance between every pair of vertices is at most t times their distance in G. It is not hard to see with our definitions that any spanning tree T will be a tree $M(T)+1$-spanner of G. Most studies of tree t-spanners have dealt with arbitrary graphs G, or at least very large classes of graphs, and have tried to minimize t (in our case, $M(T)+1$), or more commonly, have considered the complexity of algorithms that determine whether or not a tree t-spanner exists for a given input graph. In this paper, by the severe restriction of G to grid graphs, we find the smalles possible value for t, and we also find upper and lower bounds for $S(T)$, which measures the total difference in distance rather than just looking at the worst case.

2. Lower bounds

By a fairly naive argument, we can construct a lower bound for $S(T)$. Since the $m \times n$ grid graph G has $m(n-1)+n(m-1) = 2mn-m-n$ edges, and any spanning tree T of G has $mn-1$ edges, there are $mn-m-n+1 = (m-1)(n-1)$ edges of G that are not in T. Since G is bipartite, all paths between adjacent vertices have odd length, so $d_T(u,v)$ will always be even. Hence, for every $uv \in E(G) \setminus E(T)$, we have $d_T(u,v) \geq 2$. Thus, $S(T) \geq 2(m-1)(n-1)$.

This lower bound for $S(T)$ is achieved by the row-stitching pattern whenever $n \leq 3$.

We have a more interesting lower bound for $M(T)$, given in the following theorem.

Theorem 1. *Any spanning tree T of an m by n grid, with $m \geq n$, has $M(T) \geq 2\lfloor \frac{n}{2} \rfloor$.*

Proof. Let T be any spanning tree of the m by n grid. We identify the vertices of both the grid and T by their positions in terms of the rows and columns of the grid, from $(1,1)$ to (m,n). We form a walk W in T by consecutively traveling the shortest paths between vertices that appear consecutively along the edge of the grid, in a counterclockwise direction. (So we begin with the shortest path from $(1,1)$ to $(2,1)$ and end with the shortest path from $(1,2)$ to $(1,1)$.) Since W is a closed walk in a tree, every edge in T must appear an even number of times in W.

Towards a contradiction, suppose that $M(T) < 2\lfloor \frac{n}{2} \rfloor$. Since $M(T)$ is always even (the grid is bipartite), this means $M(T) \leq 2\lfloor \frac{n-2}{2} \rfloor$, so each of the shortest paths used in the construction of W has length at most $M(T)+1$; that is, at most $2\lfloor \frac{n-2}{2} \rfloor + 1$.

Consider the edges in W that pass between row i and row $i+1$ for any $\lfloor \frac{n}{2} \rfloor \leq i \leq m - \lfloor \frac{n}{2} \rfloor$. Since $i > \lfloor \frac{n-2}{2} \rfloor$, if a path from vertex $(1, j+1)$ to vertex $(1,j)$ contains an edge that passes between row i and row $i+1$ then the length of this path must be at least $2i+1 > 2\lfloor \frac{n-2}{2} \rfloor + 1$, so it is certainly not a shortest path and therefore was not used in the construction of W. Similarly, since $m-i \geq \lfloor \frac{n}{2} \rfloor$, if a path from vertex (m,j) to vertex $(m,j+1)$ contains an edge that passes between row i and row $i+1$ then the length of this path must be at least $2\lfloor \frac{n}{2} \rfloor + 1$, so again it is certainly not a shortest path and therefore was not used in the construction of W.

Hence, any edge in W that passes between row i and row $i+1$ comes from a shortest path between either a pair $(j,1)$ and $(j+1,1)$ of vertices, or a pair $(j+1,n)$ and (j,n) of vertices, for some values of j. We count the number of edges in W that pass between rows i and $i+1$ in columns $1,\ldots,\lfloor \frac{n}{2} \rfloor$, counting multiplicity. Again, since W is a closed walk in a tree, this number must be even.

Now, since $M(T) < 2\lfloor \frac{n}{2} \rfloor$, any shortest path between vertex $(j+1,n)$ and vertex (j,n) has length at most $2\lfloor \frac{n}{2} \rfloor - 1 = 2\lfloor \frac{n-2}{2} \rfloor + 1$. Hence, such a path cannot use any vertical edge that lies in any column to the left of column $n - \lfloor \frac{n-2}{2} \rfloor$. Since column $\lfloor \frac{n}{2} \rfloor$ is strictly to the left of column $n - \lfloor \frac{n-2}{2} \rfloor$, none of these paths contribute any edges to our count.

Similarly, since $M(T) < 2\lfloor \frac{n}{2} \rfloor$, any shortest path between vertex $(j,1)$ and vertex $(j+1,1)$ has length at most $2\lfloor \frac{n}{2} \rfloor - 1 = 2\lfloor \frac{n-2}{2} \rfloor + 1$. Hence, such a path cannot use any vertical edge that lies in any column to the right of column $\lfloor \frac{n-2}{2} \rfloor + 1 = \lfloor \frac{n}{2}$. So any edge between row i and row $i+1$ in such a path must be included in our count. If $j \neq i$, then the shortest path must contribute an even number of edges to our count; if $j = i$, the shortest path must contribute an odd number of edges to our count, so the total count is odd, a contradiction. □

Notice that this result immediately allows a very modest improvement to our lower bound for $S(T)$ when $n \geq 4$. That is, when $n \geq 4$, $M(T) \geq 2\lfloor \frac{n}{2} \rfloor = 4$, so for some $uv \in E(G)$, $d_T(u,v) \geq 4$. Thus, $S(T) \geq 2[(m-1)(n-1)-1]+4 = 2(m-1)(n-1)+2$. Although this is an insignificant improvement to the lower bound, it serves to show that the lower bound calculated above for $S(T)$ is never achieved when $n \geq 4$. The same argument can be extended in general to show that

$$S(T) \geq 2\left[(m-1)(n-1) - 1 + \lfloor \frac{n}{2} \rfloor\right].$$

More careful calculations using the arguments in the proof of the lower bound for $M(T)$ (specifically, the variety of choices available for the row i that is considered) may be able to achieve an improvement in the constants for this lower bound, but no success in improving the order of magnitude of the bound has been achieved.

3. Upper bounds based on constructions

We calculate the total distance $S(R)$ and the maximum distance $M(R)$ for the row pattern R (Figure 1) of an $m \times n$ grid. It is optimal to centre the "spine" in the long direction. In this case $M(R) = 2\lfloor \frac{n}{2} \rfloor$ is as small as possible (as proven in the previous section), and the total sum is given by:

$$S(R) = \begin{cases} (m-1)(n^2)/2 & \text{if } n \text{ is even} \\ (m-1)(n-1)(n+1)/2 & \text{if } n \text{ is odd} \end{cases} .$$

While the row pattern minimizes the maximum distance it does not minimize the total distance. In fact, one can replace a single edge of the tree R with a nearby edge and decrease the total sum. A pattern that reduces the total sum in the case of large grids is the **Comb Pattern**, shown in Figure 2. This pattern is similar to the row pattern in that it has a long "spine", but it differs in that instead of long "lines" attached to the spine, it has "combs". It is again optimal in terms of $S(C)$ and $M(C)$ to

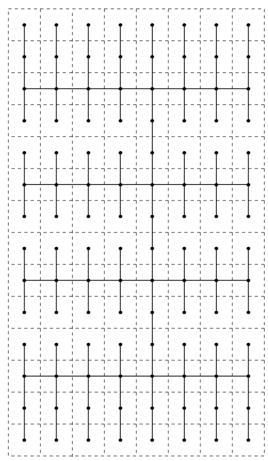

FIGURE 2. 14×8 Comb Pattern, C

centre the "spine" in the longest direction, as above. Then the maximum is $M(C) = 2\lfloor \frac{n}{2} \rfloor + 4 > M(R)$, but $S(C) < S(R)$ when n and m are sufficiently large. The values of $S(C)$ depend on the modulus of the parameters m and n, mod 3 and mod 2, respectively. Thus, there are six cases to consider; they are:

m mod 3	n odd	n even
$m \equiv 0$	$\frac{(m-3)(n-1)(n+17)}{6} + 4(n-1)$	$\frac{(m-3)(n^2+16n-16)}{6} + 4(n-1)$
$m \equiv 1$	$\frac{(m-4)(n-1)(n+17)}{6} + 8(n-1)$	$\frac{(m-4)(n^2+16n-16)}{6} + 8(n-1)$
$m \equiv 2$	$\frac{(m-5)(n-1)(n+17)}{6} + 12(n-1)$	$\frac{(m-5)(n^2+16n-16)}{6} + 12(n-1)$

Another option is to create a breadth-first search spanning tree which leads to what we call the **Breadth Pattern**, B. Let G_e be an $n \times n$ grid graph with n even and G_o be an $n \times (n-1)$ grid graph with n odd. Define an "almost square" grid graph to be a graph of the form G_e or G_o. We first define the breadth patterns B_e and B_o for these graphs. Label each vertex of the the graph by the shortest distance to the perimeter of the grid. Figure 3 illustrates the 9×8 case. For the graph G_o, there are 6 vertices labeled $(n-3)/2$ in the centre and for the graph G_e, there are 4 vertices labeled $(n-2)/2$ in the centre. Begin with a spanning tree of these centre vertices as indicated in Figure 3.

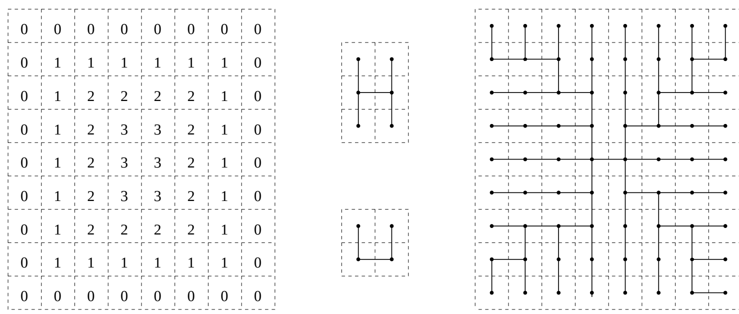

FIGURE 3. Breadth Pattern Construction

The procedure for growing the tree begins in the centre and radiates outwards. Suppose the tree has been grown to include all vertices labeled at least j, where $j > 0$. Add vertices labeled $j-1$ as described below.

- If a vertex x with label $j-1$ is adjacent to a vertex y with label j, then y is unique. Add xy to the tree.
- If a vertex x with label $j-1$ is **not** adjacent to a vertex with label j, then x has two neighbors labeled $j-1$. Arbitrarily choose one of these, say y, and add xy to the tree.

Figure 3 shows one possible breadth pattern for a 9×8 grid graph. We can easily calculate the sum of the distances for B_o and B_e in the almost square cases.

$$\begin{aligned} S(B_o) &= (n-1)(n+1)(2n-3)/6, \\ S(B_e) &= n(n+1)(n-1)/3. \end{aligned}$$

The breadth pattern, B, for any $m \times n$ grid graph is formed by placing half of the breadth pattern for the almost square grid on the top and bottom, and a row pattern in the middle. See Figure 4 for an example of this construction.

The maximum distance $M(B) = 2\lfloor \frac{n}{2} \rfloor$ again meets the lower bound, while the total sum is smaller than the row pattern sum. The values of $S(B)$ for the breadth pattern B of an $m \times n$ grid are given below.

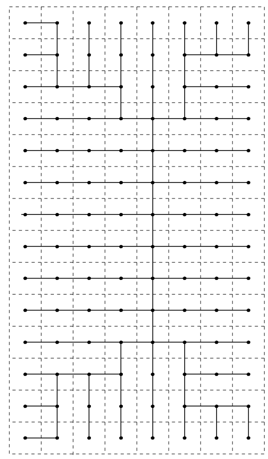

FIGURE 4. 14×8 Breadth Pattern, B

$$S(B) = \begin{cases} (3mn^2 - n^3 - 2n)/6 & \text{if } m \text{ is even,} \\ (3mn^2 - n^3 - 3m + n)/6 & \text{if } m \text{ is odd.} \end{cases}$$

Comparing these patterns with the current row pattern, we see that the comb pattern improves the total sum at the expense of an increase in the maximum distance while the the breadth pattern improves the total sum while keeping the maximum distance at a minimum. One might ask if a combination of these patterns might also be of interest. We call this new pattern a **Combed–Breadth Pattern** (Figure 5).

This does result in improvements: $S(CB)$ is the lowest value we have achieved when n and m are sufficiently large, but since its order of magnitude is still mn^2 and the formulas are complicated, we omit them.

4. Conclusions

The upper and lower bounds for $S(T)$ are quite far apart as n and m get large. It seems as though significant improvement in this direction should be possible. We are therefore some way from a proven solution to the first optimization problem mentioned, although it is our belief that the lower bound will admit of considerably more improvement than the upper bound.

We have solved the second optimization problem presented, and in fact have shown that the intuitive row-stitching tree provides a solution to this problem.

The breadth-first search tree appears to come fairly close to solving the third optimization problem presented. An inductive use of the proof of

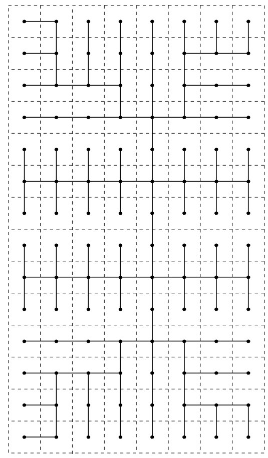

FIGURE 5. 14 × 8 Combed–Breadth Pattern, CB

Theorem 1 can be applied to show that the breadth-first search tree solves the problem of minimising the number of paths of each length, in priority order from longest paths to shortest, with $M(T)$ minimum. However, it is possible in larger examples to use an increase in the number of paths whose length is $M(T)$ (for example), to decrease $S(T)$. Hence the breadth-first search tree does not actually solve the third optimization problem.

REFERENCES

[1] Cai, L., and D. G. Corneil, Tree Spanners: an Overview, *Congressus Numerantium* **88** (1992), 65-76.

[2] Fekete, S., and J. Kremer, Tree spanners in planar graphs, *Discrete Appl. Math.* **108** (2001), 85-103.

UNIVERSITY OF ALBERTA
E-mail address: flutscher@math.ualberta.ca

THE UNIVERSITY OF MONTANA
E-mail address: mcnulty@mso.umt.edu

UNIVERSITY OF LETHBRIDGE
E-mail address: joy@cs.uleth.ca

UNIVERSITY OF CALGARY
E-mail address: kseyffar@math.ucalgary.ca