ISPRS Journal of Photogrammetry and Remote Sensing 75 (2013) 101–111

Contents lists available at SciVerse ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Global seamline networks for orthomosaic generation via local search

Steven Mills [a,*], Philip McLeod [b]

[a] Department of Computer Science, University of Otago, P.O. Box 56, Dunedin, New Zealand
[b] Areograph Ltd., P.O. Box 1077, Dunedin 9054, New Zealand

ARTICLE INFO

Article history:
Received 10 May 2012
Received in revised form 1 November 2012
Accepted 13 November 2012

Keywords:
Aerial mosaic generation
Image processing
Graph algorithms
Optimal path finding
Orthomosaic Generation
Seamline networks

ABSTRACT

A novel method for establishing a seamline network for orthomosaic generation is presented. The seamline network determines the regions of the mosaic allocated to each of the original images. Our method allows for optimal network vertices to be selected while avoiding a combinatorial optimisation task. The computational requirements (time and storage) of our proposed method are linear in the number of images. Seamlines between vertices are found using a graph-based approach which finds shortest paths, subject to a constraint which minimises the maximum cost of any single edge. Experiments on synthetic and real-world data sets illustrate the value of the approach compared to naïve vertex selection and recently proposed methods.

1. Introduction

Creating composite images from aerial photographs is a key step in orthomosaic generation. With the advent of inexpensive, high quality, digital imaging; the increasing use of unmanned aerial vehicles (UAVs); and the development of digital photogrammetry techniques, orthomosaics are being made from large sets of low-altitude aerial images. This means that more images need to be composed to cover a given area, and that parallax effects are more significant. As a result there has been significant attention paid to automated techniques for generating orthomosaics from aerial images.

A typical processing pipeline for automated orthomosaic generation can be summarised as follows:

1. (Digital) images are captured, generally along with navigation data—usually the GPS location of the camera at the time each image was captured, but possibly also camera orientation from inertial sensors.
2. Structure-from-motion techniques are applied to generate a terrain model, along with the camera locations and orientations. Alternatively an existing terrain model can be used (Mills et al., 2009).
3. The images are reprojected onto the terrain model, creating an orthoimage for each input photograph.
4. These images are mosaiced in such a way as to minimise the visual transition from one image to the next.

In this paper we concentrate on the final step of this process. We assume that we are provided with orthoimages in some common co-ordinate frame, and our task is to create a single orthomosaic of the area covered. Similar image mosaicing problems arise in a variety of contexts, and there are many sub-problems and algorithms available, see Szeliski (2006) for a survey.

In the final mosaic image, each source image contributes data to a small region. The boundaries between these regions are called seamlines. The problem of finding a seamline between a pair of images has been well studied (Fernandez et al., 1998; Kerschner, 2001; Zomet et al., 2006; Botterill et al., 2010; Chon et al., 2010; Pan and Wang, 2011; Yang et al., 2011; Wan, 2012).

These methods typically minimise some cost function over the difference between the overlapping images along the seamline. Chon et al. (2010), for example, first identify the maximum intensity difference that must be crossed by any seamline, and then use this to remove any points with greater difference from consideration. A graph search is then used to find a minimum cost path through the remaining region of the image. This finds a path that minimises the worst error (maximum difference) at any single pixel, and among all such paths finds the one with the minimum total error.

A simple image difference is generally used for the cost function, but more detailed methods have been proposed. Pan and Wang (2011) use image gradients as well as the image difference to avoid crossing edge features. Yu et al. (2012) propose a more

* Corresponding author.
E-mail addresses: steven@cs.otago.ac.nz (S. Mills), phil@areograph.com (P. McLeod).

0924-2716/$ - see front matter © 2012 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS) Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.isprsjprs.2012.11.003

complicated cost function, based on measures of colour difference, edge features, texture, proximity to the nadir point, and feature saliency. These are combined in a weighted sum to give an overall cost for each pixel in the image. The final seamline is then found as a minimum cost path through this image.

These techniques can be applied recursively to compose images into a mosaic. The mosaic is initialised as a single image, and then each subsequent image is added with an appropriate seamline. The final set of seamlines separating the image regions is known as the seamline network.

This recursive approach, however, is dependent on the order in which images are composed. In some applications, such as real-time mosaicing from video (Botterill et al., 2010), there is a natural ordering to the images. In general, however, an order-independent solution is desirable to ensure that a globally optimal solution can be found.

Yang et al. (2011) describe a bisector based seamline algorithm. The overlapping regions between image pairs is approximated by a quadrangle, and a bisecting polyline is used as the seam. This seam placement method is generalised to more complex polygons and multiple images, but seamline placement is made without reference to the image contents. As a result there is no effort made to avoid areas where the seamline will cause significant artefacts.

Pan et al. (2009) present a system for automatically generating a seamline network in a global sense. They introduce the concept of 'area Voronoi diagrams with overlap' (AVDO), where the starting point for the seamline network assigns each pixel to the nearest image centre (by the Manhattan distance). In order to simplify this process they assume that the individual orthoimages are well approximated by quadrangles, and that the intersection of a pair of overlapping images is also a quadrangle. The first assumption is often a good approximation, but the second does not hold in general, as illustrated in Fig. 1.

Once an initial seamline network has been generated it is refined in two steps. Both steps are based on minimising the difference between co-incident pixels from the source images—by placing the seamlines in regions where the adjacent images have minimal difference, the visibility of the seamline is minimised. Firstly the network junctions (where three or more seamlines meet) are placed where the maximum difference between the (three or more) overlapping images is minimal. Secondly a bottleneck model (Fernandez et al., 1998) is used to find paths between the seamlines that minimise the maximum difference along the path. Since the network junctions are fixed, each path can be independently estimated.

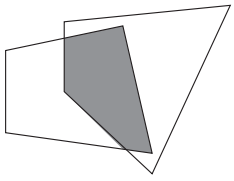

Fig. 1. The intersection of two quadrangles is not always a quadrangle.

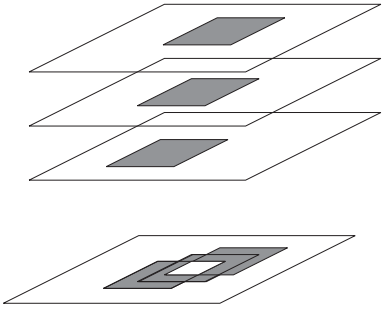

Fig. 2. A misaligned feature (shaded square) in three images (top), leads to a region of low difference (bottom) surrounded by a region of high difference (shaded). While the central region looks like a good candidate for placing a junction, any paths reaching it must pass through the high difference region.

The method for network junction positioning may, however, select sub-optimal locations. Fig. 2 illustrates how a point which has minimal difference could be surrounded by an area of high difference. Any paths to the junction must then incur the cost of passing through this high difference area. Searching for optimal junction locations, however, risks losing the local nature of the seamline optimisation. If a junction is moved, then the seamlines connecting it to its neighbouring junctions change, which could cause the neighbours' optimal locations to change, and so forth. Unless this can be prevented, a combinatorial optimisation problem arises over the set of all possible junction locations, and an exhaustive search becomes computationally infeasible.

Our proposed method is similar to that of Pan et al. (2009), but makes the following contributions:

- We remove the assumption that the input images' extents are well approximated by axis-aligned rectangles. This assumption is not true in general, particularly with the ability of many UAVs to fly arbitrarily defined GPS paths.
- We replace the heuristic (fixed) placement of the seamline network junctions by a local search for optimal positions. This provides greater flexibility in selecting optimal seamlines.
- We provide a method for finding globally optimal seamlines through a set of local optimisation procedures. This means that the proposed method scales linearly with the number of images, and is well suited to distributed processing.

The optimisation algorithm for seamline paths used is based on Chon et al. (2010). We first identify a bottleneck point, which is a point with the highest unavoidable cost—any seamline must pass through a point with a cost at least as high as the bottleneck, but there is at least one path that avoids any higher cost points. A minimum cost path is then found through the bottleneck via graph search. However, we replace Chon et al.'s (2010) recursive bottleneck estimation procedure with a single-pass approach based on the graph's minimal spanning tree.

2. Seamline placement and optimisation

Our proposed method begins with an initial estimate of the seamline network in the mosaic image. The requirements on this estimate are minimal—the initial seamlines must lie in the overlapping regions of the images they join, and each image must be surrounded by a closed curve of seamlines except at the edges of the mosaic. We represent this network as a seamline graph $G_S = (V_S, E_S)$, with vertex set $V_S = \{v_1, v_2, \ldots, v_n\}$, and edge set $E_S = \{e_{ij} | v_i \text{ and } v_j \text{ are linked by a seamline}\}$.

The initial seamline networks used in our research are based on a Voronoi diagram (Aurenhammer, 1991). The camera location for each image (known from GPS) is projected on to a ground plane. A Voronoi diagram is then made, allocating each point in the mosaic to the nearest camera centre. The boundaries between Voronoi cells are our initial seamline network, and vertices are formed where three or more cells meet. The details of the initial seamline estimation network, however, are not critical, and our proposed approach can be used to refine seamlines estimated by any method.

Next, a set of search regions is established. Firstly, a vertex search region, R_i, is defined around each $v_i \in V_S$, defining the area that will be searched for an optimal junction location. These re-

gions must be continuous and include v_i. Furthermore, all pairs of regions must be disjoint and non-adjacent. That is, there must be at least one pixel between each pair of distinct vertex search regions. In addition all points in R_i must lie within the extent of every image joined by a seamline incident on v_i.

Secondly, an edge search region, R_{ij}, is defined around each $e_{ij} \in E_S$. R_{ij} must be adjacent to, but disjoint from, R_i and R_j. Furthermore it must be continuous and lie entirely within the intersection of the two images adjacent to the seamline. These search regions, and the associated notation, are illustrated in Fig. 3. Note that in the figure the vertex search regions are circular and the seamlines and edge search regions are linear. Neither of these attributes are required by our algorithm, however these are convenient choices for implementation.

Our method then proceeds as follows:

1. We define a set of graphs to represent the task of finding optimal seamlines (Section 2.1).
2. We then find a bottleneck point, b_{ij} in each edge search region, R_{ij} (Section 2.2).
3. Finally we optimise over each vertex search region, R_i, to find the location that gives the minimum total cost over the paths to all adjacent bottlenecks (Section 2.3).

2.1. Graph construction

Having defined the search regions we need a way to evaluate seamlines. This is done by forming a weighted graph, the cost graph, $G_{C_{ij}} = (V_{C_{ij}}, E_{C_{ij}})$, for each seamline e_{ij}. The vertices, edges, and edge weights are defined with reference to the mosaic image pixel locations, (x,y). The vertices of the graph are the corners of the pixels; edges link corners which are vertically or horizontally adjacent; and edges are weighted by the sum of image differences at the pixels on either side of the edge.

More formally, each pixel location, (x,y) can be interpreted as being a region in the image (the pixel itself), or a corner co-ordinate, which we take to be the lower right corner as shown in Fig. 4. Note that we use the usual image-based co-ordinates with x increasing from left to right and y increasing from top to bottom, and use corner-based locations so that the seamlines run between rather than through pixels. We now define the vertex and edge sets of $G_{C_{ij}}$ as

$$V_{C_{ij}} = \{(x,y) | (x,y) \in R_{ij}\}, \tag{1}$$

$$E_{C_{ij}} = \{((x,y),(x-1,y)) | (x,y),(x-1,y) \in V_{C_{ij}}\} \cup \{((x,y),(x,y-1)) | (x,y),(x,y-1) \in V_{C_{ij}}\}. \tag{2}$$

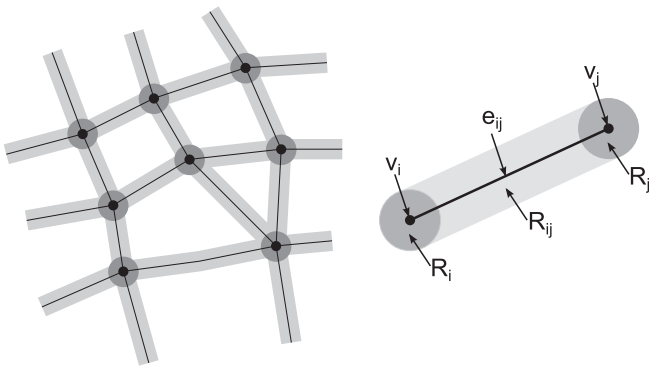

Fig. 3. A vertex search region (dark grey) is defined for each junction/vertex (black circles), and an edge search region (light grey) for each seamline/edge (black lines). The vertices, edges, and search regions are labelled as shown on the right for the edge from v_i to v_j.

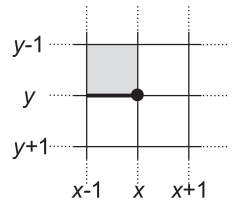

Fig. 4. Pixel locations (x,y) can be interpreted as a region in the image (shaded) or a corner co-ordinate (circle).

This graph represents the possible paths through an edge search region R_{ij}. The graph's vertex set is the set of pixels in the edge search region, and each vertex is connected to its four immediate neighbours.

The weights of the edges are computed from the two images, I and J, connected by the seamline e_{ij} as the sum of the image differences at the two pixels adjacent to the edge. For example, the weight of the horizontal bold edge shown in Fig. 4 would be computed as

$$w((x,y),(x-1,y)) = d(x,y) + d(x,y+1), \tag{3}$$

where $d(x,y)$ is some non-negative function that computes a difference between pixel values at $I(x,y)$ and $J(x,y)$. Likewise, the weight for a vertical edge would be given by

$$w((x,y),(x,y-1)) = d(x,y) + d(x+1,y). \tag{4}$$

A simple and common choice for d is the absolute difference in intensity, but other metrics such as the Euclidean distance in some colour space (such as RGB or HSV), or more advanced perceptual difference measures could be used (Yu et al., 2012; Luo et al., 2001). Factors other than pixel colour values could also be incorporated. For example, the cost function could be chosen to avoid pixels near the edge of images (which suffer from greater lens distortion and chromatic aberration), or to favour near-nadir views over oblique views.

We also define an expanded cost graph, $G^+_{C_{ij}}$ in the same manner over the larger vertex set

$$V^+_{C_{ij}} = R_{ij} \cup R_i \cup R_j. \tag{5}$$

This expanded graph represents all possible paths from a point in a vertex search region R_i, passing through an edge search region R_{ij}, to some point in another vertex search region R_j.

2.2. Bottleneck detection

Having defined the graphs above, we now consider each edge search region, R_{ij}, in turn. We want to find seams through this region that minimise the maximum weight edge in the cost graph $G_{C_{ij}}$. This *minimax* edge is the bottleneck of the seamline, and is found as follows:

1. We begin with the expanded cost graph, $G^+_{C_{ij}}$.
2. We set to 0 the weight of all edges in $E^+_{C_{ij}} \setminus E_{C_{ij}}$ (i.e. that are in R_i or R_j rather than R_{ij}).
3. We find the minimal spanning tree (MST) of this graph. Prim's or Kruskal's algorithms can be used for this task (Cheriton and Tarjan, 1976).
4. The bottleneck, b_{ij}, is now the maximum weight edge of the path (there can be only one—see below) through the MST from any point in R_i to any point in R_j.

Setting the weights to zero within the vertex search regions means that there is no cost associated with moving within a vertex search region, R_i, to its boundary with an adjacent edge search re-

gion R_{ij}. At least one such boundary point must exist and be reachable from any point in R_i, since both R_i and R_{ij} are continuous, and R_{ij} is constructed so that it is adjacent to R_i. This means that from any point in R_i we can move to any boundary point with R_{ij} at no cost, traverse R_{ij} to a boundary point with R_j and then to any point within R_j at no cost. Once the MST is found, we can find a path from any point in R_i to any point in R_j, let b_{ij} be the edge on this path with the maximum weight. Since this path lies within the MST it is a minimax path (its maximum weight edge has the minimal possible weight), and it is possible to move within R_i and R_j at no cost, so any path from R_i to R_j must pass through an edge with at weight at least as large as that of b_{ij}.

2.3. Junction location optimisation

Having identified the bottleneck points in each edge search region, each junction can be optimised over the vertex search regions. Consider a possible junction location, $v \in V_i$. Rather than finding an optimal path from this vertex to each neighbouring vertex search region, V_j, we can restrict ourselves to finding a path to the corresponding bottleneck, b_{ij}, as illustrated in Fig. 5. The remainder of the path to V_j is found in a separate optimisation over paths to b_{ij} from vertices $v' \in V_j$. The result of this is that we can treat each junction location as a separate optimisation problem, avoiding a combinatorial optimisation over the possible locations for all junctions at once.

Suppose that the current vertex under consideration is V_i, and is linked by edges to $V_1, V_2, \ldots, V_k$. We wish to find an optimal junction position, $v_i \in V_i$, where there are good seamlines to each of $b_{i1}, b_{i2}, \ldots, b_ik$.

Given any $v \in V_i$ we can find a good seamline to a bottleneck b_{ij} as follows:

1. Begin with the expanded cost graph G^+_{ij}.
2. Remove any edges $e \in E_{C_{ij}}$ such that $w(e) > w(b_{ij})$. The resulting graph is still connected (since it preserves the previously computed MST over $E_{C_{ij}}$, and all edges in $E^+_{C_{ij}} \setminus E_{C_{ij}}$).
3. Find (using Dijkstra's algorithm (Dijkstra, 1959) or similar) the shortest path from v to b_{ij}.

The resulting path avoids all edges with weight greater than the bottleneck in the edge search region, and has the lowest total weight of all such paths. We can compute the paths from the bottleneck to all $v \in V_i$ with a single application of Dijkstra's algorithm, giving the shortest paths to all candidate junction locations.

Let $p(v, b_{ij})$ be the shortest path from $v \in V_i$ to b_{ij}, and let $\omega(p(v, b_{ij}))$ be the total weight of this path (the sum of the weights of its edges). We can then find an optimal junction location as

$$v_0 = \arg\min_{v \in V_i} \sum_{j=1}^{k} \omega(p(v, b_{ij})). \quad (6)$$

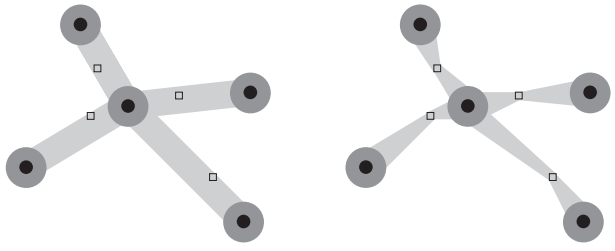

Fig. 5. Having detected bottlenecks (black squares) in edge search regions, we can change our search from possible paths between adjacent junctions (left) to searching from each region to adjacent bottlenecks (right).

This approach works for internal edges—when both of the vertices lie within the mosaiced region. At the boundaries of the mosaic, however, seamlines extend to (or beyond) the edges of the input images. One approach to deal with this is to replace the bottleneck point with a point outside of the two images which meet at the seamline. A shortest path can then be found from the junction vertex to this target point, with no cost incurred for passing through the region outside of the input images.

3. Experimental results

In the following experiments the initial seamline network is computed as a Voronoi diagram of the camera locations projected onto the ground plane. This favours views that are near-nadir, and avoids oblique views wherever possible. Vertex search regions are defined as circular regions of a fixed radius, r. To avoid adjacent regions, Voronoi vertices located less that $2r + 1$ pixels apart are merged before the search regions are defined. This merging ensures that the vertex search regions are disjoint, and that there is at least 1 pixel space between them for the edge search regions. Edge search regions are defined by a region of width r around a straight line linking the centres of the vertex search regions.

While these do not in general guarantee that the image overlap requirements are met, in the cases tested these are generally upheld. In situations where the search region contains small areas outside one of the imaged areas, the relevant edges are removed so that seamlines are always placed in valid locations. This effectively reduces the initially defined search regions to their largest valid sub-regions. As mentioned previously, alternative methods for initialisation could be used, such as Pan et al.'s (2009) AVDO regions.

The algorithms have been implemented in C++ in Visual Studio 2010. OpenCV (Bradski, 2000) is used for image processing operations; QHull's `qvoronoi` utility (Barber et al., 1996) is used for Voronoi diagram computation; and Boost's graph library (Siek et al., 2002) for minimal spanning tree and shortest path estimation. All experiments and timings were conducted under 64-bit Windows 7 running on an iMac with a quad-core 2.7 GHz Intel Core i5 processor and 8 GB of RAM. The current implementation is single threaded.

3.1. Synthetic imagery

In order to test the algorithms under controlled conditions, a set of synthetic test scenes are made. These are created from 1200×1200 pixel rectangles cut from a 5362×5357 orthomosaic image of the city of Davis, California. This image is one of a set from http://www2.dcn.org/orgs/orthophotos/. The resulting tiles overlap by 200 pixels in each direction and are offset by a small random shift to create misalignments in the mosaicing. Fig. 6 shows the original image with the tile boundaries marked.

In order to illustrate the value of searching for optimal vertex locations, three different methods are compared. All three place the vertices within the search regions, and then find minimum cost paths to all of the adjacent edge bottlenecks. The methods can then be evaluated by the total cost of the paths from the selected vertex location to the bottlenecks. The three methods are:

1. Place the vertex at the centre of the search region (no vertex search).
2. Place the vertex at the location within the search region where the difference between the overlapping images is minimal (Pan et al., 2009).
3. Place the vertex at the location within the search region where the sum of the path costs is minimal (proposed method).

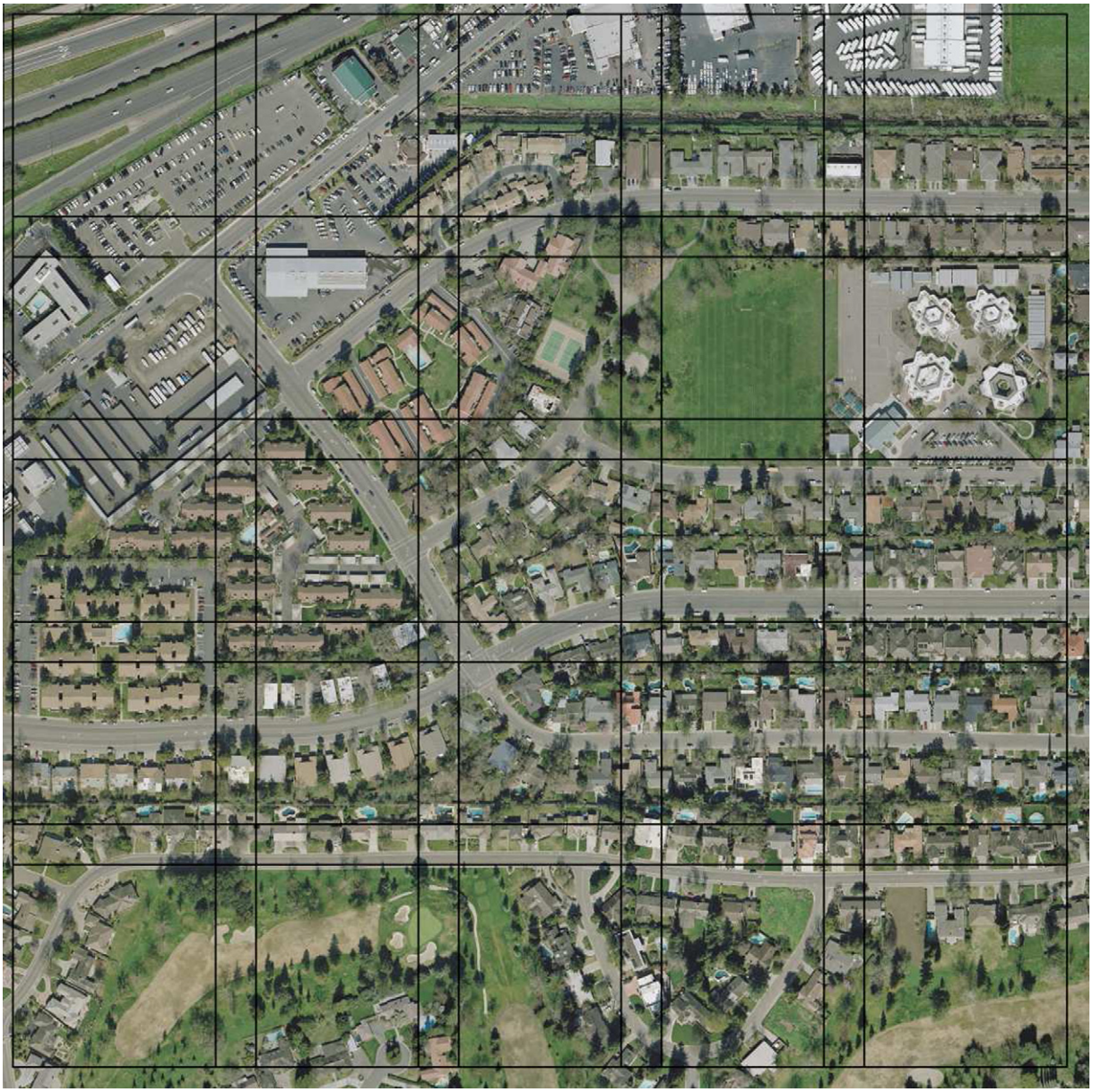

Fig. 6. Tiles from an orthomosaic image of Davis, used to create synthetic test cases.

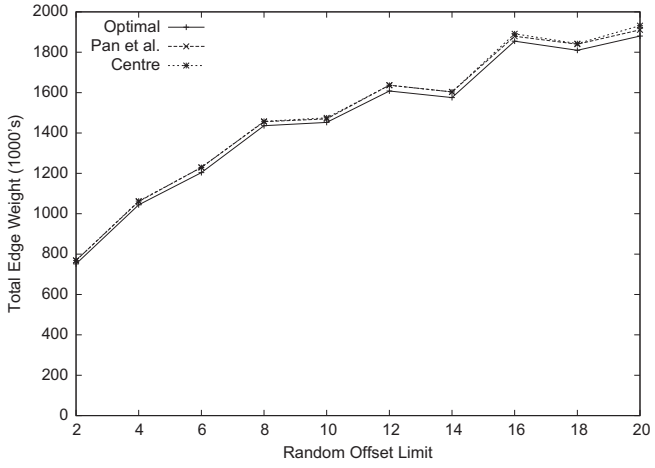

Fig. 7. Total edge weight of the seamline network for the three methods with varying degrees of random misalignment.

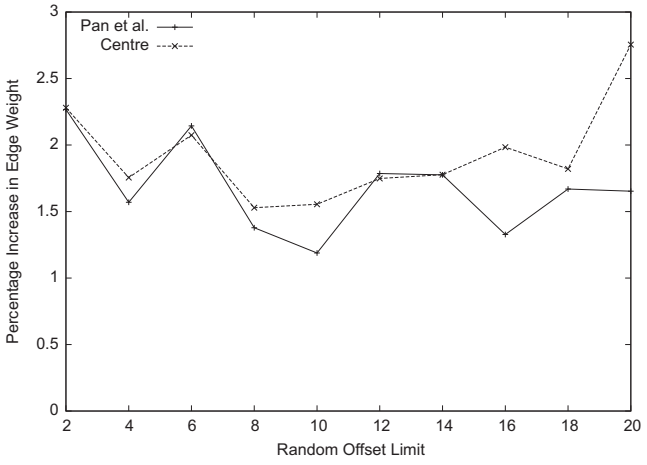

Fig. 8. Percentage increase in edge weight across the seamline network for Centre vertex placement and the method proposed by Pan et al. over the Optimal placement for varying degrees of random misalignments.

Fig. 9. Selected regions of (from left to right) the original image and the results of seamline finding using the Centre of the search region; the method of Pan et al. (2009); and the proposed Optimal method to find seamlines.

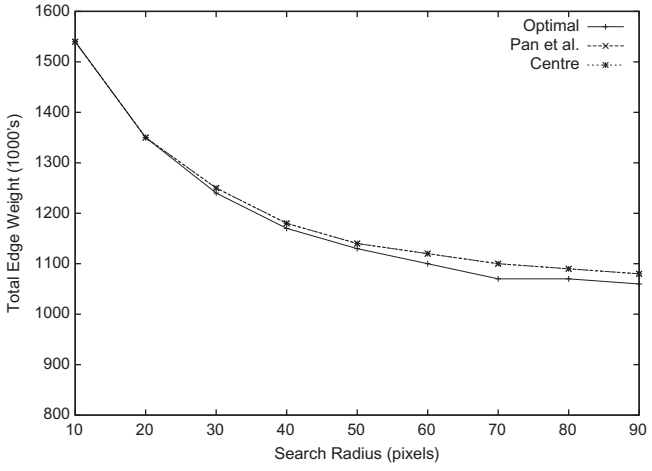

Fig. 10. Total edge weight of the seamline network with varying search region radius.

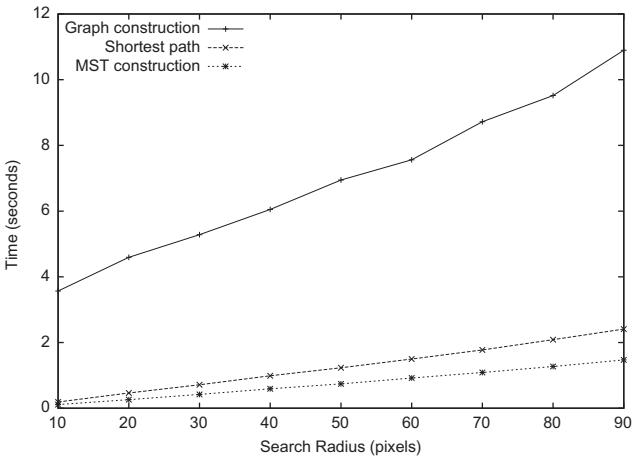

Fig. 11. Time required for graph operations with varying search region radius.

We will refer to these as the *Centre*, *Pan et al.*, and *Optimal* methods respectively.

Clearly the Optimal method will have the lowest score, but the comparison with the Centre and Pan et al. methods serves to illustrate the value of the proposed approach. Fig. 7 shows the total graph edge weights of the seamline network for different maximum offset distances. The offset in the x and y directions for each trial was randomly chosen from a uniform distribution up to the maximum offset, which ranged from 2 to 20 in steps of 2. The ra-

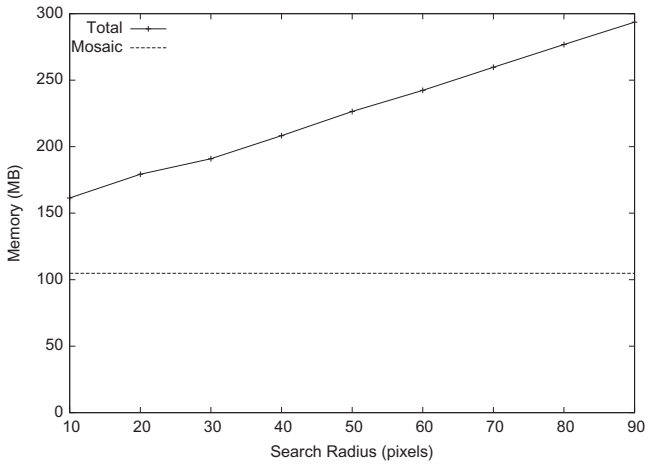

Fig. 12. Memory required with varying search region radius.

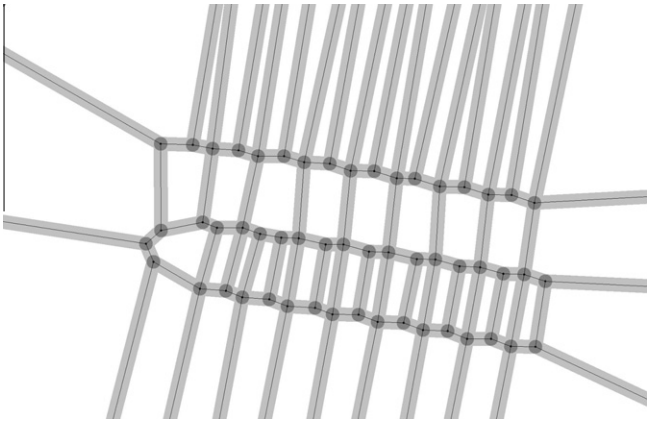

Fig. 14. Voronoi-based search regions for the Rangiora suburban dataset.

dius for the vertex and edge search regions was fixed in this experiment at $r = 50$. A clear trend of increasing total edge weight (i.e. more visible seamlines) with increasing offset is visible, which is expected as greater offsets leads to more significant image misalignment.

The other trend visible in Fig. 7 is that the vertex placement heuristic suggested by Pan et al. (2009) gives similar results to placing the vertices at the centres of the search regions, and both are noticeably worse than the locations found by our method. This can be seen more clearly by plotting the percentage increase in total edge weights over the Optimal vertex placement, as shown in Fig. 8. This is computed as

$$\% \text{ increase} = 100\% \times \frac{W - W_O}{W_O}, \tag{7}$$

where W is the total weight from the method being considered, and W_O is the total weight with Optimal vertex placement. Pan et al.'s method usually gives lower total path costs than Centre placement, but not in all cases.

Qualitative evidence of the improvement given by the proposed algorithm can be seen in Fig. 9. This figure shows regions of the mosaic where there are significant visual differences between the three methods' results. The original image is shown for comparison, but the examples are taken from the trial where the input tiles are offset by up to 20 pixels in the x and y directions, and so there is often no perfect seamline to select. In example (a) of Fig. 9 none of the methods are able to reconstruct the misaligned road correctly, but Pan et al. and the Optimal method have reconstructed the building more accurately than the Centre vertex placement. In example (b) none of the methods has given an ideal reconstruction of the building structure, while in example (c) the Centre weighted approach has produced multiple images of the vehicle beside the trees. In examples (d) and (e) both Centre placement and Pan et al. have produced discontinuities in the building outlines which are avoided by the Optimal vertex placement.

The trials in this example took on average 16.9 s, with little variation in time with the degree of misalignment. There was little difference between the three method's timings—Pan et al.'s (2009) was 0.02 s slower than the Optimal method on average due to slightly higher costs associated with vertex selection, while the

Fig. 13. Sample image from the Rangiora suburban dataset.

Fig. 15. Final mosaic for the Rangiora data set using our optimal solution.

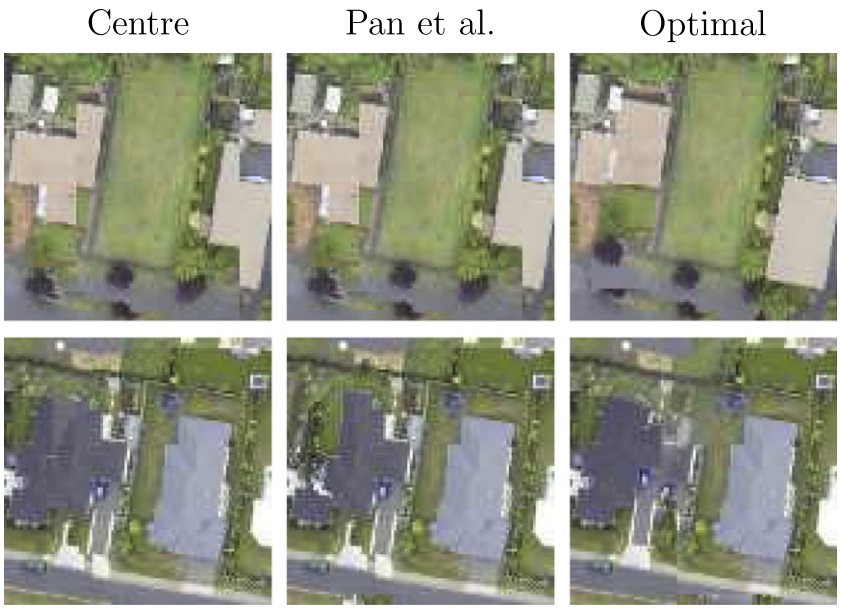

Fig. 16. Details from the mosaics for the Rangiora data set using (from left to right) the original Centre points, Pan et al.'s (2009) heuristic placement, and our Optimal solution.

Centre method which avoids such computation completely was slightly (0.04 s) faster. The largest amount of time (6.7 s) was used constructing the graphs from the images, and rendering the final mosaics took around 2.6 s in each case. Creating the Voronoi diagram took 0.03 s, computing the minimal spanning tree 0.73 s, and finding optimal paths via Dijkstra's Algorithm 1.2 s.

The quality of the paths also depends on the search radius, as shown in Fig. 10. Increasing the search radius means that a wider variety of paths can be explored, which generally decreases the path weights. Note, however, that this decrease is not guaranteed. A larger search region might mean that a lower weight bottleneck can be chosen. In some cases this could lead to a higher total edge

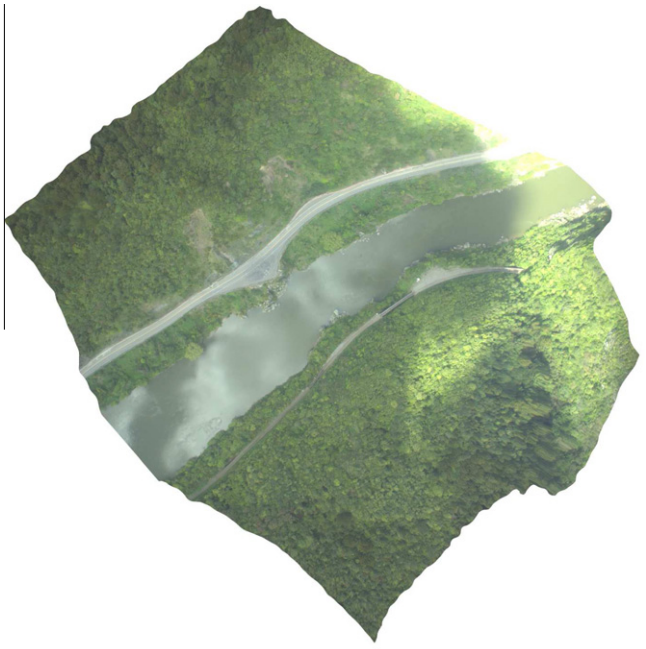

Fig. 17. Sample image from the Manawatu Gorge data set. Note the lighting variation due to passing clouds, and the reprojection artefacts in the lower right of the image.

weight due to a longer path in terms of number of edges. This 'longer' path, however, would have a lower maximum edge weight, and so be less visually apparent.

Fig. 10 also shows diminishing returns in terms of reduced path weight as the search radius increases. Increasing the search radius also comes at a cost as the graphs become larger and so the time and memory requirements of the processing increases. A larger value of r allows more paths to be explored, but never removes any paths from consideration. As a result, increasing r will never decrease the quality of the paths (as measured by the minimax path with the lowest total path cost), and a value of r large enough to cover the entire overlap between an image pair will choose the

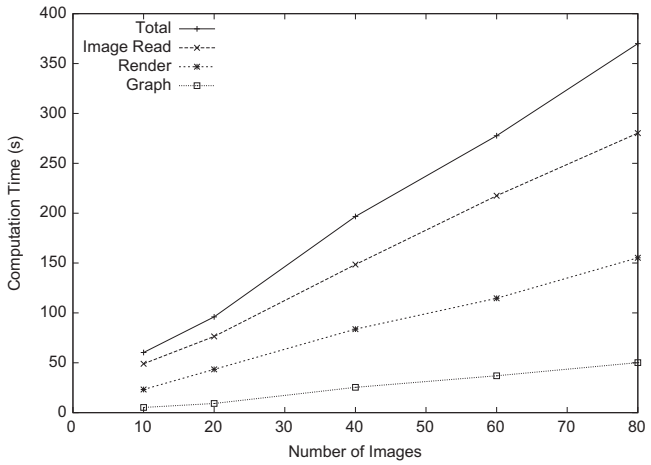

Fig. 19. Time required to process scenes with a varying number of images. Times are shown for the entire process, image reading, graph construction, and mosaic image rendering.

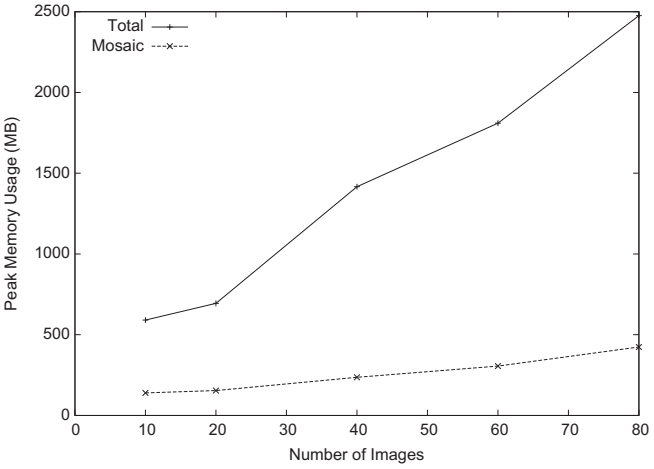

Fig. 20. Memory required to process scenes with a varying number of images. The memory needed to store the mosaic image is shown for comparison.

Fig. 18. Final mosaic for the Manawatu data set using Optimal vertex placement (Method 3).

best path from among all possible paths. However, as r increases the time and memory requirements increase proportionally, while providing smaller and smaller increases in the mosaic quality.

The time required for constructing and searching the graphs depends on the number of pixels, p, in the search regions. Graph construction is $O(p)$, while the algorithms to find the MST and shortest paths are $O(p \log p)$. The number of pixels in the search region depends on the length of the Voronoi edges and the width of the search regions. Fig. 11 shows a plot of the time in seconds required for graph construction, MST construction, and shortest path detection for varying search region radii. In these cases the Voronoi edges are of a fixed length (1000 pixels), so p is proportional to r.

In this experiment the linear cost of graph construction dominates the time requirements. This suggests that unless very large search regions are used the actual cost will be roughly linear in the search region size.

The memory requirements also depend on p, and therefore r. The largest consumers of memory are the graph structures and the mosaic image itself. The size of the graph structure, however, is proportional to p, while the mosaic size is constant for a given task. Fig. 12 shows the peak memory required and that used for the mosaic image with varying search radius.

3.2. Rangiora suburban area

The first real example is a set of 54 images of Rangiora, a township north of Christchurch, New Zealand. The input orthoimages are rendered at 0.5 m/pixel and show significant misalignments. The final mosaic image is 1885×1216 pixels.

Note that the quality of these orthoimages, both in terms of ground resolution and alignment, is significantly worse than typical results from image-based terrain model generation and alignment. These large errors, however, serve to illustrate the method and highlight the differences between different approaches. A sample image from this data set is shown in Fig. 13.

An initial seamline network is made by starting with a Voronoi tessellation and merging adjacent vertices. Search regions for vertex and edge placement are then defined using the fixed radius ($r = 20$ pixels) as shown in Fig. 14.

Pan et al.'s (2009) method gave a 10.4% higher total path cost across the mosaic than the Optimal method. Using the Centres without any optimisation gave an 9.9% higher total path cost than the Optimal vertex placement, and so was slightly better than Pan et al.'s (2009) approach in this example. Fig. 15 shows the final mosaic produced by the proposed method.

The poor alignment and large intensity variation between images means that none of the algorithms can produce an artefact-free mosaic. A closer look at some of the regions, however, shows that the use of Optimal vertex locations does provide for qualitatively better seams in many cases, as well as giving a quantitative improvement. Examples of this are shown in Fig. 16. In the first example, none of the methods has found an ideal seam for the road to the bottom, but only the Optimal vertex placement has given plausible structure for the two large buildings adjacent to the field. In the second example, the building to the left is best reconstructed by Optimal vertex placement.

The total processing time for this example was 14.9 s on average, with the method of Pan et al. (2009) being slightly (0.11 s) slower and the Centre vertex placement slightly faster (0.03 s) than the Optimal vertex placement algorithm. Graph construction took 9.2 s, and rendering the final mosaic took 1.5 s. The minimal spanning tree took 0.06 s to construct, while finding shortest paths took 0.23 s. Processing required 70.4 MB of memory, of which 8.7 MB was used for the final mosaic.

3.3. Manawatu gorge

The final example presented here is a set of 80 images from the Manawatu Gorge in the lower North Island of New Zealand. While these images are well aligned, there are intensity variations due to vignetting, lighting changes, and exposure differences. The terrain in this regions is quite steep as well as having local height variation due to tree lines and forest. As a result, the orthomosaics also show some artefacts from the reprojection process, as shown in Fig. 17. The final mosaic shows the good alignment of the images, but the effects of vignetting can be seen in the river area, and the change in exposure and lighting can be clearly seen across the length of the mosaic. These effects can be corrected for, but are not the focus of the current investigation.

The mosaics are rendered at a resolution of 0.2 m/pixel, and the final mosaic image is $10{,}939 \times 7620$ pixels. A search radius of $r = 15$ was used. The final mosaics are visually very similar, and the mosaic with Optimal vertex locations is shown in Fig. 18.

Although the mosaics produced using Central vertices and Pan et al.'s (2009) method appear nearly identical to the Optimal vertex locations, there are subtle differences. The total path length using central vertices was 1.32% higher than the optimal solution, while that found by Pan et al.'s method was 1.20% higher. Processing for this example took 373 s on average, with the mosaic renderings taking 155 s and graph construction 50.2 s. The Voronoi graph construction, graph searches for shortest paths, and minimal spanning trees look only 2.25 s in total. Reading the images into memory took a significant proportion of the time in this case, at 280 s (note that the graph construction and mosaic rendering times include some image reading).

We also use this example to illustrate the change in processing requirements with the number of images. The algorithms were run on subsets of the images containing 20, 40, 60, and 80 images. The time required for processing is shown in Fig. 19 along with the times required for the most expensive parts of the process (graph construction, image read time, and mosaic rendering for comparison). It can be seen that the time required is roughly linear in the number of images, which is expected. The most time consuming aspects of the algorithm are image reading, mosaic rendering, and graph construction, which are linear in the number of images. While other parts of the process (Voronoi diagram, MST extraction, and shortest path computation) are $O(n \log n)$, n these processes are fast in practice, and so do not contribute greatly to the running time. Furthermore, n for the MST and shortest path estimation problems is determined by the length of individual seamlines, and so does not generally increase as additional images are added.

The memory requirements also vary with the number of images, as shown in Fig. 20. The total memory required is significantly greater than that needed to store the mosaic image. This is almost all used by the graph structures, which are currently held in memory throughout the process. For larger scenes the memory requirements may exceed the available RAM on desktop machines. In these cases the graph structures could easily be cached on disk, since their construction processes are independent of one another, and their use in determining optimal paths and vertex locations is a local operation, using only a few graphs per vertex.

4. Conclusion

We have presented a method for Optimal vertex placement for seamline network generation. These vertices produce lower cost seamlines than either naïve vertex placement at the centre of the search region, or the method recently proposed by Pan et al. (2009). The bottleneck detection method used is similar to that of Chon et al. (2010), but uses minimum spanning trees rather than

an iterative approach to determine the bottleneck location. The examples shown here demonstrate a qualitative improvement in mosaic quality, well as balancing the minimisation of the graph path weights in terms of the highest single edge weight and the total path weight.

A naïve approach to optimising the vertex locations could risk a combinatorial optimisation problem, as altering one vertex's position could have flow-on effects through the entire network. We avoid this problem by identifying bottlenecks between vertex search regions, and the resulting method takes time linear in the number of images to be mosaiced. Although some parts of the algorithm are $O(n \log n)$ they are very fast in practice, and so the $O(n)$ terms with larger constant factors dominate. The storage requirements have also been shown to increase linearly with the number of images.

The current implementation uses simple methods for seamline initialisation (Voronoi diagrams), search region definition (constant radius regions), and graph weight estimation (intensity differences). The techniques presented here, however, do not depend on any of these choices. More advanced algorithms could be applied in any of these areas, such as the graph weighting strategy proposed by Yu et al. (2012), which includes aspects such as texture, feature saliency, and location relative to the nadir point in the graph weighting. Other tools for improving mosaicing, such as exposure compensation and moving object removal (Uyttendaele et al., 2001; Mills and Dudek, 2009) could also be applied to improve the final orthomosaic.

Acknowledgments

The authors would like to thank Hawkeye UAV Ltd. for providing the 'Rangiora' and 'Manawatu' data sets used in this paper. We would also like to thank the City of Davis for making their orthophotos freely available from http://www2.dcn.org/orgs/orthophotos/.

References

Aurenhammer, F., 1991. Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Computing Surveys 23 (3), 345–405.

Barber, C.B., Dobkin, D.P., Huhdanpaa, H.T., 1996. The Quickhull algorithm for convex hulls. ACM Transactions on Mathematical Software 22 (4), 469–483, <http://www.qhull.org/> (accessed 13.11.12).

Botterill, T., Mills, S., Green, R., 2010. Real-time aerial image mosaicing. In: Proc. Image and Vision Computing New Zealand, pp. 1–8.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools. <http://opencv.willowgarage.com/> (accessed 13.11.12).

Cheriton, D., Tarjan, R.E., 1976. Finding minimum spanning trees. SIAM Journal of Computing 5 (4), 724–742.

Chon, J., Kim, H., Lin, C.-S., 2010. Seam-line determination for image mosaicking: a technique minimizing the maximum local mismatch and the global cost. ISPRS Journal of Photogrammetry and Remote Sensing 65 (1), 86–92.

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numerische Mathematik 1, 269–271.

Fernandez, E., Garfinkle, R., Arbiol, R., 1998. Mosaicking of aerial photographic maps via seams defined by bottleneck shortest paths. Operations Research 46 (3), 293–304.

Kerschner, M., 2001. Seamline detection in colour orthoimage mosaicking by use of twin snakes. ISPRS Journal of Photogrammetry and Remote Sensing 56 (1), 53–64.

Luo, M.R., Cui, G., Rigg, B., 2001. The development of the CIE 2000 colour-difference formula: CIEDE2000. Color Research & Application 26 (5), 340–350.

Mills, A., Dudek, G., 2009. Image stitching with dynamic elements. Image and Vision Computing 27 (10), 1593–1602.

Mills, S., Park, D., Pinchin, J., Barnsdale, K., Hide, C., 2009. Integrating GNSS, IMU, and imagery for automatic orthomosiac generation. In: ION GNSS, pp. 442–452.

Pan, J., Wang, M., 2011. A seam-line optimized method based on difference image and gradient image. In: International Conference on Geoinformatics, pp. 1–6.

Pan, J., Wang, M., Li, D., Li, J., 2009. Automatic generation of seamline network using area Voronoi diagrams with overlap. IEEE Transactions on Geoscience and Remote Sensing 47 (6), 1737–1744.

Siek, J.G., Lee, L.-Q., Lumsdaine, A., 2002. The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley. <http://www.boost.org/doc/libs/release/libs/graph/> (accessed 13.11.12).

Szeliski, R., 2006. Image alignment and stitching: a tutorial. Foundations and Trends in Computer Graphics and Vision 2 (1), 1–104.

Uyttendaele, M., Eden, A., Szeliski, R., 2001. Eliminating ghosting and exposure artifacts in image mosaics. In: IEEE Conf. on Computer Vision and Pattern Recognition, vol. II. pp. 509–516.

Wan, Y., 2012. Tracking of vector roads for the determination of seams in aerial image mosaics. IEEE Geoscience and Remote Sensing Letters 9 (3), 328–332.

Yang, Y., Guo, Y., Li, H., Han, Y., 2011. An algorithm for remote sensing image mosaic based on valid area. In: International Symposium on Image and Data Fusion, pp. 1–4.

Yu, L., Holden, E.-J., Dentith, M.C., Zhang, H., 2012. Towards the automatic selection of optimal seam line locations when merging optical remote-sensing images. International Journal of Remote Sensing 33 (4), 1000–1014.

Zomet, A., Levin, A., Peleg, S., Weiss, Y., 2006. Seamless image stitching by minimizing false edges. IEEE Transactions on Image Processing 15 (4), 969–977.