Real-time aerial image mosaicing

Tom Botterill[1], Steven Mills[2], Richard Green[3]

[1]Geospatial Research Centre, University of Canterbury, Christchurch 8041, NZ.
[2]Areograph Ltd, 90 Crawford St, Dunedin, NZ.
[3]Department of Computer Science, University of Canterbury, Christchurch 8041, NZ.
Email: tom.botterill@grcnz.com

Abstract

This paper describes a scheme for seamlessly stitching together images captured from an aerial platform, in real-time, in order to provide an operator with a larger field-of-view. Both recent images, and images from earlier in a flight are used. To obtain real-time performance several of the latest computer vision techniques are applied: firstly the Bag-of-Words image representation allows overlapping images to be found efficiently, and provides cheap wide-baseline correspondences between them. Secondly the BaySAC robust estimation framework allows images to be registered efficiently from a prior motion model combined with large numbers of potential matches between cheap image patch descriptors. Thirdly an efficient seam-placement algorithm allows the rendering of a visually attractive mosaic. Results are presented on a sequence of high-resolution images captured from a microlight.

Keywords: Image mosaicing, image stitching, Bag-of-Words, BaySAC, wide baseline registration

1 Introduction

Image mosaicing is the process of joining overlapping images together to form a larger image, hence enlarging a cameras' field-of-view. One application for this is to increase the field-of-view of a camera streaming images back from an aerial platform, such as an aeroplane or UAV (Unmanned Aerial Vehicle), without the additional cost, and in the case of a UAV lens weight and/or bandwidth requirements of capturing larger images in the first place. This aids an operator by keeping features of interest in view for longer.

This paper describes our scheme for real-time mosaicing of aerial images. High-resolution images are stitched together around the latest image, providing a seamless wide-angle view of the surrounding area, in real-time. Three innovations make this possible: firstly actively searching for adjoining images using the Bag-of-Words (BoW) algorithm, enabling the use of both recent images and images from earlier in the flight. Secondly, perspective transformations between images are computed by combining correspondences provided by BoW with the BaySAC framework, which allows efficient outlier removal while incorporating a prior motion model where available. Thirdly, an efficient shortest-path algorithm that finds visually unobtrusive seams between frames enables a seamless mosaic to be rendered. No previous scheme has had this capability to generate large seamless mosaics in real-time.

This paper is organised as follows: Section 2 describes the latest contemporary mosaicing techniques and their limitations; Section 3 describes our more efficient approach, Section 4 presents experimental results; and Section 5 discusses these results.

2 Background

There has been extensive research into image mosaicing; the review papers by Szeliski [1] and Zitová and Flusser [2] describe this literature in detail. Many of these methods however make assumptions that are not true for aerial images, or are computationally expensive batch processes for producing massive Google-Earth-style mosaics, so in this section we will concentrate on those techniques that are most likely to be suitable for real-time mosaicing of aerial imagery.

Zitová and Flusser [2] identify four stages that most image mosaicing and registration processes have in common. These are:

1. **Feature detection** Salient elements of each image are identified and located.
2. **Feature matching** Correspondences between features are established, typically by comparing feature descriptors.

978-1-4244-9631-0/10/$26.00 2010 IEEE

3. **Transform estimation** The correspondence between features is used to determine a transform that maps one image to the other.

4. **Image Composition** The images are transformed and aligned with one another. Some form of interpolation is often applied to blend the images.

In general to align two images of the same scene the images must be mapped onto a model of the 3D surface shown. Examples of this include the offline processes used to generate mosaics used in applications such as Google Earth[1] or Microsoft Photosynth[2], and incremental procedures which are not currently practical in real-time, but eventually aim to produce mosaic-based 3D maps of a robot's environment, such as the underwater mapping scheme by [3]. For images from an aerial platform however, we can generally assume that the ground is approximately planar, i.e. it is viewed from sufficient distance that relative depth variation is low and parallax is small. This allows a much simpler model to be used: two images of a planar surface captured through a pinhole camera are related by a perspective transformation: if two images each show the same plane, then there is a homography (3×3 matrix) H such that for all $\mathbf{x}$ in the first image and $\mathbf{x}'$ in the second image (represented in homogeneous coordinates), $H\mathbf{x} = \mathbf{x}'$.

For many mosaicing applications a simpler model is appropriate: for a rotating camera at a fixed location (all points are on the plane at infinity), a simple 2D translation and 1D rotation is sufficient to align two images. For a strictly down-pointing camera on an aerial platform a similarity transform (rotation, translation and scale) is sufficient. For a camera with a small field-of-view the perspective transformation may be approximated with an affine transformation: $A\mathbf{x} + \mathbf{t} = \mathbf{x}'$ for a 2×2 transformation matrix A and 2D translation $\mathbf{t}$.

2.0.1 Feature detection and matching

The first stage in estimating the transformation between two images is to find matches between features in one image and the same features in the other image. These features are usually points within each image chosen to be localisable and repeatable, such as Harris Corners [4] or SIFT's Difference-of-Gaussian blobs [5]. A descriptor vector describing the area around each feature is then extracted, for example a SIFT feature or just a simple image patch [1]. Possible matches between these descriptors are then found, for example by pairwise comparison.

[1] `http://earth.google.com`
[2] `http://photosynth.net`

2.0.2 Transform estimation

Given at least four point correspondences we can estimate a perspective transform via the Discrete Linear Transform, or DLT [6, Section 4.1]. This least squares method reduces the effects of noise in the image measurements. Least-squares solutions are, however, susceptible to being corrupted by outliers, which are common amongst correspondences due to repeated similar-looking features and because of moving objects in the scene. To overcome this Random Sample and Consensus (RANSAC) approaches are commonly used [7]. RANSAC works by repeatedly selecting small random subsets of correspondences (hypothesis sets) from which to compute candidate solutions. Each candidate solution is compared with the entire data set until a solution compatible with a large number of correspondences is found. These are assumed to be inliers, and a least-squares solution is computed from this inlier set.

One mosaicing scheme closely following this pattern is the scheme described by [8] for producing panoramic photographs. SIFT features are used to firstly to identify which frames overlap, then RANSAC is used to compute perspective transforms between these overlapping images. Image boundaries are blended together with a multi-scale blur to smooth edges in areas of low detail, while preserving structure in areas of higher detail. While results are visually impressive, performance is far from real-time, taking several minutes to mosaic 10 images.

Similar schemes include the system by [9], which matches SURF features between microscope slides to produce one image of a larger sample, however despite the simple motion model this still requires tens of seconds per frame; and the system by [10] which registers batches of images using RANSAC, producing impressive globally consistent mosaics, although not in real-time, partly due to the expensive optimisation needed to ensure globally consistent transforms.

An alternative to feature-based image matching is to directly align the images to one another, by using correlation across entire images to estimate the transformation between them. Nielsen et al. [17] use a Fourier transform to calculate the correlation between images from a camera spinning about the vertical axis, and render a cylindrical mosaic at 30Hz, however this simple model has predictable motion with only one DOF (the camera orientation), and generates a mosaic of fixed size. Similarly [18] find similarity transforms between pairs of aerial images by brute-force search, however this takes 88 seconds per 256×256 frame due to direct alignment's complexity exponential in the number

of parameters in the model being fitted, and hence we do not consider a direct approach feasible for a real-time system where perspective effects are significant.

As is the case of visual SLAM, features can be tracked from one frame to the next, for example in the scheme described by [19], which stitches together images captured by a stationary rotating camera to produce a spherical mosaic. A Kalman filter and a model of the camera rotation predicts the region of a future frame in which each tracked feature will lie. This approach works well at high frame-rates, however at lower frame-rates or during rapid image motion large regions must be searched for each feature, making this approach considerably more costly and error-prone than a wide-baseline approach, and in addition a wide-baseline approach is still needed if images from earlier passes are to be registered. In aerial images large image motion often occurs due to camera rotation, and it is more desirable to use available bandwidth for boosting the total area imaged, rather than for transmitting large numbers of almost-entirely overlapping images, making tracking unsuitable for our application.

2.0.3 Image composition

Once transforms between images have been computed a mosaic can be rendered by choosing one image, then warping every other image to the same coordinate frame. However the two images will rarely line up exactly, due to a combination of small errors remaining in the transform estimate, uncorrected lens distortion, and intensity variation between images. In the case of aerial images the fact that the ground is not exactly planar adds to this misalignment. A naïve approach to composition, laying each image over the existing mosaic results in visible and potentially distracting artefacts due to straight edges in the image (such as roads and boundaries) being broken at the join, and the new straight edges appearing along borders (Figure 1). Two common solutions to this problem are firstly to hide seams by interpolating between overlapping images, or secondly to choose seams minimising visual discontinuities.

A variation of the first approach is used by [8], which successfully hides seams, although moving objects or poor alignment can result in multiple images of objects appearing ('ghosting'). A similar approach could be feasible in real-time. Alternatively Levin, Zomet, et al. [11, 12] estimate a final mosaic by minimising a cost function based on image gradient similarities, however at several minutes per-frame this would not be feasible for a real-time application.

The second approach is to place a seam between images chosen to be visually inconspicuous. This seam should cut natural edges in the images at places where they line up (or ideally not at all if possible). Various methods for finding optimal graph-cuts minimising dissimilarity functions between pixels [13, 14] or segments [15] have been proposed, however these methods are currently far-from practical in real-time. An alternative approach is to use Dijkstra's algorithm to find the seam (path) where the difference between the two images is minimised [16]; this efficient approach is feasible for real-time processing.

In summary, while numerous mosaicing schemes have been proposed, most are targeted at producing extensive, globally consistent mosaics and are unsuitable for real-time processing. Those systems that do run in real time assume camera motion models inappropriate for aerial images.

3 Video mosaicing in real-time

The previous section described contemporary approaches to image mosaicing, and their limitations. In this section we describe our new scheme for real-time aerial image mosaicing, RT-AIM, which follows the same basic pattern as many existing schemes, but uses the latest computer vision algorithms to enable real-time performance.

The following sections describe the individual components of RT-AIM: firstly Section 3.1 describes the BoW image representation and how it is used both to find adjoining frames, and to find correspondences between images; Section 3.2 describes how these correspondences are used for image registration; and Section 3.3 describes how seams between registered images are found, and how we efficiently render a seamless mosaic. Full source code, and an example video, are available online at `http://hilandtom.com/tombotterill`.

3.1 Feature description and the Bag-of-Words image representation

To extract image features in real-time we use simple descriptors based on image patches centred on corners from the FAST corner detector [20]. The highest-scoring FAST corners are chosen subject to a minimum separation constraint (to ensure geometry is well conditioned). For large, high-resolution images patches sampled from across large patches of the image provide more distinctiveness than smaller patches; typically patches sized 66×66 pixels are down-sampled to 11×11. Extracting FAST corners from a 800×600 image takes 30 milliseconds, and extracting 450 patch descriptors and adding them to a hierarchical BoW database takes typically 4 milliseconds per frame, allowing every frame to be

indexed (Figure 2). In order to register images from earlier passes, when features were viewed at different orientations, patches are rotated so that the intensity gradient of each patch is in the same direction, in a similar manner to SURF descriptors [21]. In aerial images from approximately down-pointing cameras, the changes in perspective and scale between images is usually small, so perspective-invariant descriptors such as SURF or SIFT are not necessary, and anyway are too computationally expensive to extract from large images in real-time [20, 22].

The BoW algorithm represents each image as a set of descriptors that it contains. Each descriptor is quantised to the nearest of a set of representative descriptors (a 'dictionary' of 'image words') that they contain, enabling fast comparisons between pairs of images–images with many 'image words' in common are likely to show the same place. The BoW image representation is widely used to aid robot positioning by 'active loop-closure detection'; i.e. for detecting when a robot visits somewhere it knows. For mosaicing we use it to recognise when two images have substantial overlap, so that they may be registered.

The BoW implementation used is based on the scheme described by [23]. This uses a hierarchical dictionary allowing fast indexing of images, and a heuristic similarity measure based on word occurrence statistics. In addition a speeded-up approximate clustering scheme allows the dictionary to be re-created on-the-fly, ensuring the dictionary is representative of the images viewed without any need for prior training [24].

Once images are represented as a BoW, we can find the most similar matches from earlier in the flight. To find all available images in the vicinity of the latest frame we recurse over the images similar to these matches (including the previous frame). We then attempt to compute a transformation with each of the top few matches. Note that finding overlapping images using a BoW database is hugely faster than approaches based on matching descriptors between all image pairs (for example [8]), and no navigation data is necessary to find these matches.

3.1.1 Correspondences from the Bag-of-Words algorithm

A feature visible in two frames normally results in each frames' BoW representation containing the same corresponding image word. Usually there are several of a particular word in two images; all possible pairs can be considered as candidate matches. In this case we directly compare all pairs of descriptors, and take all pairs where each descriptor matches the other significantly more closely than it matches any other. When a set of N descriptors appears similar to many (M) in the other image all possible matches are considered as potential correspondences. These '$N - M$ correspondences' are common in 'self-similar' environments with repeating structures. Finding correspondences in this way is very cheap, taking around 8ms per pair of frames, but typically over 80% of the matches found by a considerably slower brute-force search are found.

3.2 Computing relative positions

For aerial images we assume that the ground is approximately planar, and hence images can be aligned with a perspective transformation (the affine approximation is not sufficiently accurate—alignment errors of about 5% are typical). Other schemes use the RANSAC framework and DLT to estimate transformations (Section 2.0.2), however RANSAC performs poorly when outlier rates amongst correspondences are high, as it cannot take into account prior information about correspondences' reliability. Instead of RANSAC we use the BaySAC framework [25]. In BaySAC each hypothesis set chosen is the one most likely to contain only inliers, based on data points' prior inlier probabilities and conditional on the hypothesis sets that have failed to lead to good models (presumably because they were contaminated by outliers). The advantages of this approach are firstly to find the correct model in fewer iterations, and secondly to find inliers amongst a large number of low-quality $N - M$ matches without degrading performance. Large numbers of inliers are desirable to minimise errors caused by small or poorly-conditioned inlier sets. This is important when mosaicing aerial images as there is often only a small overlap between pairs of consecutive images; this leads to initially high outlier rates, and often poorly conditioned geometry when matched features lie only in a narrow strip of the image.

To assign prior probabilities to correspondences the following assumptions are made: firstly, without any knowledge of the camera's motion, the prior probabilities of each point in an image correctly matching any point in the other image are uniform (with probability p). Therefore if point i in one image is matched to M_i points in the other image, the probability of each of the M_i candidate correspondences being in the inlier set $\mathbb{I}$ is $P(i \in \mathbb{I}) = p/M_i$, and these probabilities are disjoint (each point can be matched to only one other).

Secondly, when we are registering consecutive frames (at times t and $t+1$) we can use information from the transformation found between frames $t-1$ and t, $H_{t-1,t}$ to predict the locations of each feature in image $t+1$. The assumption here is that the velocity at time t is a good initial estimate for the

velocity at time $t+1$; a good assumption for fixed-wing aerial platforms. In addition we can estimate the acceleration of image features[3]. Probabilities of correspondences being inliers can now be updated to incorporate the following motion model using Bayes theorem: given a correspondence between two points $i = (\mathbf{x}_i, \mathbf{x}'_i)$ we assume that if i is an inlier, $\mathbf{x}'_i$ is distributed normally about $H_{t-1,t}\mathbf{x}_i$ with standard deviation s pixels (about 150 pixels works well), and if i is an outlier $\mathbf{x}'_i$ is distributed uniformly over the image. This gives i the following p.d.f.:

$$\begin{aligned} f(i) &= f((\mathbf{x}_i, \mathbf{x}'_i)) \qquad (1) \\ &= \begin{cases} \phi_{(0,s^2)}(\mathbf{x}'_i - H_{t-1,t}\mathbf{x}_i) & i \in \mathbb{I} \\ 1/A & i \notin \mathbb{I} \end{cases} \end{aligned}$$

where ϕ is the 2D normal p.d.f. with each component i.i.d. with mean 0 and variance s^2, and A is the image area.

$$\begin{aligned} & P(i \in \mathbb{I}|H_{t-1,t}) \qquad (2) \\ = & \frac{f(i|i \in \mathbb{I})P(i \in \mathbb{I})}{P(i \in \mathbb{I})f(i|i \in \mathbb{I}) + P(i \notin \mathbb{I})f(i|i \notin \mathbb{I})} \end{aligned}$$

When $N - M$ correspondences are used, the fact that each point in one image matches to at most one in the other must also be considered:

$$\begin{aligned} & P(j \in \mathbb{I}|j \text{ incompatible with i}) \qquad (3) \\ = & P(j \in \mathbb{I})\frac{1 - P(i \in \mathbb{I}|H_{t-1,t})}{1 - P(i \in \mathbb{I})} \end{aligned}$$

The perspective transformation from BaySAC is further refined using top-down outlier-removal [26]: a transformation is fitted to all inlier correspondences, then the error in each correspondence is recomputed, allowing more inliers to be found while potential outliers are removed. This is iterated several times. Note that the BoW database will occasionally find incorrect matches, but in this case BaySAC will fail to find a transformation, allowing the rejection of these matches.

3.3 Rendering seamless mosaics

We have implemented three alternative strategies for image composition: firstly images are simply warped into the frame of a single image, producing a mosaic very rapidly but with obvious seams remaining. Secondly a simple interpolation scheme combines overlapping images with weights in proportion to each pixel's distance from the frame's centre. Thirdly we find cuts between pairs of images that minimise the total squared difference between the two frames along the cut, using Dijkstra's algorithm, as described by [16]. This third option is most successful at eliminating artefacts in both man-made and natural environments (Figure 1); however a search over all paths through the image is costly. Instead we sub-sample the area to be searched and find a path at this courser resolution. This path is usually visually acceptable, especially at higher frame-rates when it is only briefly visible.

[3]As with single camera incremental structure-from-motion this is not quite the same as modeling the acceleration of the camera, as image feature motion depends on the plane's altitude as well as its velocity, so an uninformative value should be used that suitable for any height the plane is likely to fly.

3.4 Optimisations for real-time operation

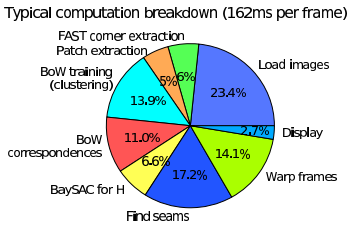


Figure 2: Breakdown of execution time—one 1000 × 1000 mosaic containing an average of 6.8 frames is rendered and displayed per-frame.

A typical breakdown of computation times is shown in Figure 2; times are computed on a desktop computer with a 3GHz Intel Core 2 Duo processor. Once optimised no particular areas dominate, however many optimisations were necessary to achieve this performance, in particular to speed-up image warping. To warp one image into another, we predict where the source image will map to in the destination image, then iterate over an appropriate area in the destination image, calculating where in the source image each pixel should come from. Only the nearest source pixel is considered. To make this work in real-time the following optimisations were necessary: firstly, while in general a perspective transform is computed, often the elements $H_{3,1}$ and $H_{3,2}$ are close to zero (i.e. the transform is approximately affine), and are exactly zero for one image in each rendered mosaic. In this case a slightly faster affine warp is applied. Secondly, if $H_{3,1}$ is not too large, the division needed to compute the source pixel location can be approximated with a binomial expansion: each source pixel location is given by $(x'_s, y'_s) = (x_s/t_s, y_s/t_s)$ where $(x_s, y_s, t_s) = H^{-1}(x_d, y_d, 1)$. Given t_s^{-1} at (x_d, y_d), t'^{-1}_s at $(x_d + 1, y_d)$ is given by

$$\begin{aligned} t'^{-1}_s &= ((H^{-1})_{3,1}(x+1) + (H^{-1})_{3,2}y + 1)^{-1} \\ &= t_s^{-1}(1 + (H^{-1})_{3,1}t_s^{-1})^{-1} \\ &\approx t_s^{-1}(1 - (H^{-1})_{3,1}t_s^{-1}) \qquad (4) \end{aligned}$$

(a) No seam removal (b) Interpolation between images (c) Optimal seam between images

Figure 1: Simply warping images over each other leads to visible seams and broken edged when images do not quite align (a). The seams are successfully hidden by interpolating between frames, but artefacts around man-made straight edges still occur (b). An optimal seam between images usually removes both kinds of artefact (c).

Thirdly, four-channel images are used, so that each three-byte RGB triple is aligned with a 32-bit word, allowing faster copying of pixels throughout the program despite the increase in memory required. It may be possible to improve performance using SIMD instructions or to carry out some processing on the GPU, however no single optimisation would greatly affect the performance.

The only parts of out scheme with complexity increasing with time are querying the BoW database, and re-building the dictionary. BoW queries have complexity linear in the number of images, although in practice the cost of these queries is negligible for up-to tens of thousands of images. Re-building the dictionary has complexity log-linear in the number of images, however this is only necessary while the environment keeps changing in appearance. Eventually a dictionary suitable for all environments likely to be encountered will be found.

4 Results

We have tested RT-AIM with an aerial image sequence from a down-pointing Canon 400D camera attached to a microlight. 3220 images sized 800×532 and covering farmland, forest, coastline and urban areas were captured at 2.8Hz, then processed sequentially to produce a video-mosaic sized 1200×1000 (Figure 3). One mosaic is generated for every frame, and a frame-rate of 6.2 Hz is maintained, with each mosaic containing an average of 8 images, often including frames from earlier in the flight.

Table 1 shows that BaySAC with a simple prior probability model is no more successful than RANSAC at finding transformations, but finds considerably larger inlier sets, as it makes use of $N - M$ correspondences. However when a prior motion model (MM) is used, BaySAC clearly outperforms RANSAC, enabling transformations to be found 80% of the time, compared with 69% of the time for RANSAC. The motion model particularly helps during rapid motion when image overlap is small and a low proportion of potential correspondences are correct. As a result mosaics rendered when BaySAC is used contain an average of 8 frames, compared with 5.2 frames when using RANSAC.

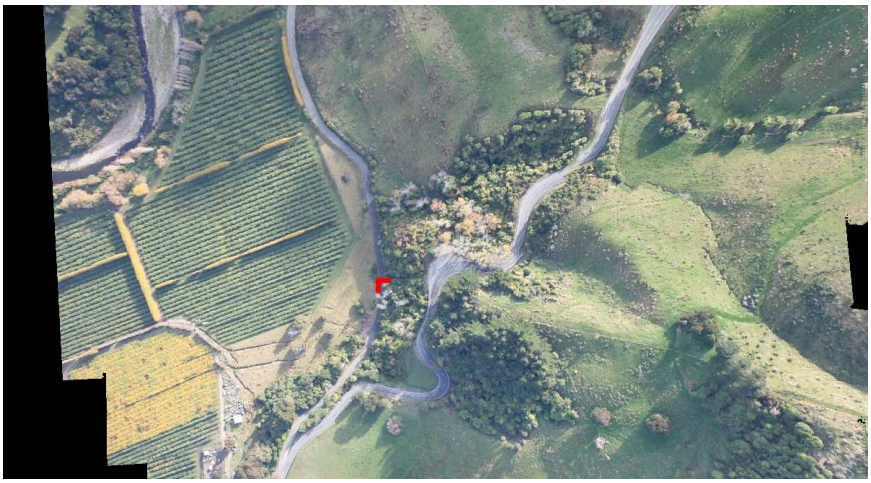
For higher frame-rate video, images sized 320×212 can be mosaiced at 20Hz. Alternatively a larger 1440×2000 video-mosaic, filling two computer monitors, is rendered at 3.6Hz, still faster than images were captured. Towards the corners of this mosaic however images are warped with accumulated sequences of several transformations, and become distorted in places. In future we will investigate optimising transformations across multiple images in order to prevent this.

5 Conclusions

A mosaicing scheme has been described that enables the mosaicing of images captured from an aerial platform, hence providing a larger field-of-view for the operator. Images from throughout the flight are registered and stitched together seamlessly in real-time. This real-time performance is enabled by the following four key implementation choices: firstly expensive invariant descriptors are not necessary for aerial images; simple oriented image patches chosen around FAST corners provide good distinctiveness at a fraction of the cost. Secondly, in order to robustly match sufficient numbers of features between frames, the BaySAC framework is used, which reduces costs compared to conventional RANSAC and enables an accurate perspective transformation to be found from large numbers of poor-quality correspondences. Thirdly the BoW algorithm allows images near to the current frame to be found from throughout the flight, and in addition provides cheap wide-baseline correspondences between pairs of frames. Finally, an efficient optimal cut-finding algorithm applied to sub-

(a)

(b)



(c)

Figure 3: Mosaicing succeeds in a wide range of environments, including difficult visually sparse and self-similar regions. (b) and (c) include images from earlier passes, found by BoW.

Table 1: Success at finding transformations between image pairs from a stream of 3220 images with different parametrisations, terminating after 250 iterations.

Method	Av. # iterations	% Success	# inliers on success
BaySAC+MM	172	80%	67
BaySAC	193	69%	73
RANSAC+$N-M$	245	23%	76
RANSAC+no $N-M$	209	69%	53

sampled images, together with an optimised image warping procedure allow large mosaics to be rendered and displayed at 6Hz, over twice the framerate at which images were captured.

References

[1] Szeliski, R.: Image alignment and stitching: A tutorial. Technical Report MSR-TR-2004-92, Microsoft Research (2006)

[2] Zitová, B., Flusser, J.: Image registration methods: A survey. Image and Vision Computing **21** (2003) 977–1000

[3] Johnson-Roberson, M., Pizarro, O., Williams, S.B., Mahon, I.: Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. J. Field Robotics **27** (2010) 21–51

[4] Harris, C., Stephens, M.: A combined corner and edge detection. In: Proc. The Fourth Alvey Vision Conference. (1988) 147–151

[5] Lowe, D.G.: Object recognition from local scale-invariant features. In: ICCV (1999)

[6] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press (2003)

[7] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM **24** (1981)

[8] Brown, M., Lowe, D.G.: Automatic panoramic image stitching using invariant features. Int. J. Computer Vision **74** (2007) 59–73

[9] Rong, W., Chen, H., Liu, J., Xu, Y., Haeusler, R.: Mosaicing of microscope images based on surf. In: Proc. IVCNZ (2009)

[10] Turkbeyler, E., Harris, C.: Building aerial mosaics II. In: Proc. Electromagnetic Remote Sensing Defence Technology Conf. (2009)

[11] Levin, A., Zomet, A., Peleg, S., Weiss, Y.: Seamless image stitching in the gradient domain. In: Proc. ECCV (2004) 377–389

[12] Zomet, A., Levin, A., Peleg, S., Weiss, Y.: Seamless image stitching by minimizing false edges. Trans. Image Process **15** (2006)

[13] Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Trans. Graphics **23** (2004)

[14] Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts? Trans. PAMI **26** (2004) 147–159

[15] Gracias, N., Gleason, A., Negahdaripour, S.: Fast image blending using watersheds and graph cuts. In: Proc. BMVC (2006) 469–478

[16] Davis, J.: Mosiacs of scenes with moving objects. In: Proc. CVPR. (1998) 354–360

[17] Nielsen, F., Andre, A., Tajima, S.: Real-time spherical videos from a fast rotating camera. In: ICIAR (2008) 326–335

[18] Tegolo, D., Valenti, C.: A naïve approach to compose aerial images in a mosaic fashion. In: Proc. ICIAP (2001) 512–516

[19] Civera, J., Davison, A.J., Magalln, J.A., Montiel, J.M.M.: Drift-free real-time sequential mosaicing. Int. J. CV **81** (2009) 128–137

[20] Rosten, E., Drummond, T.: Machine leaning for high-speed corner detection. ECCV (2006)

[21] Tuytelaars, T., Gool, L.V.: Matching widely separated views based on affine invariant regions. Int. J. Computer Vision **59** (2004)

[22] Cummins, M., Newman, P.: Highly scalable appearance-only SLAM - FAB-MAP 2.0. In: Proc. Robotics: Science and Systems (2009)

[23] Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: CVPR (2006)

[24] Botterill, T., Mills, S., Green, R.: Speeded-up Bag-of-Words algorithm for robot localisation through scene recognition. In: IVCNZ (2008)

[25] Botterill, T., Mills, S., Green, R.: New conditional sampling strategies for speeded-up RANSAC. In: Proc. BMVC (2009)

[26] Rousseeuw, P.J., Leroy, A.M.: Robust regression and outlier detection. Wiley (1987)