Real-Time Spherical Mosaicing using Whole Image Alignment

Steven Lovegrove and Andrew J. Davison

Imperial College London, London SW7 2AZ, UK
{sl203,ajd}@doc.ic.ac.uk

Abstract. When a purely rotating camera observes a general scene, overlapping views are related by a parallax-free warp which can be estimated by direct image alignment methods that iterate to optimise photoconsistency. However, building globally consistent mosaics from video has usually been tackled as an off-line task, while sequential methods suitable for real-time implementation have often suffered from long-term drift. In this paper we present a high performance real-time video mosaicing algorithm based on parallel image alignment via ESM (Efficient Second-order Minimisation) and global optimisation of a map of keyframes over the whole viewsphere. We present real-time results for drift-free camera rotation tracking and globally consistent spherical mosaicing from a variety of cameras in real scenes, demonstrating high global accuracy and the ability to track very rapid rotation while maintaining solid 30Hz operation. We also show that automatic camera calibration refinement can be straightforwardly built into our framework.

Key words: real-time tracking, spherical mosaicing, SLAM, auto-calibration

1 Introduction

A set of images can be fused into a mosaic if there is no parallax between them, and this is the case either when a generally moving camera browses a plane or when a general 3D scene is observed by a camera which only rotates. There is a great deal of literature on building mosaics from multiple images or video (see the tutorial by Szeliski [1]). The emphasis has been on methods which operate off-line, consisting of pair-wise image registration achieved either with features (e.g. [2] using SIFT matching, or [3]) or whole image alignment (e.g. [4]), and global optimisation. Meanwhile, methods that were able to operate from video in real-time such as [5] achieved accurate local registration but were subject to drift over longer periods due to the lack of explicit global optimisation.

The core issue of mosaicing is to accurately estimate the motion of the camera, and if globally consistent mosaics are to be constructed from video in real-time this motion estimation must be drift-free over arbitrarily long time periods. Like any case of estimating the motion of an outward-looking sensor in a previously unknown environment, mosaicing can be considered as a Simultaneous Localisation and Mapping (SLAM) problem. This is important, because in

SLAM research, originating in the mobile robotics area, there has been great attention paid to developing algorithms which run sequentially in real-time but are also able to generate globally consistent scene models.

The predominant early approaches to SLAM were based on sequential probabilistic filtering algorithms, most importantly the Extended Kalman Filter (EKF), to jointly estimate the positions of both the moving sensor and the features which it observed. This methodology was recently successfully applied to image mosaicing by Civera *et al.* [6], in the first work which was able to demonstrate drift-free mosaicing at frame-rate from a rotating camera. The computational cost of the EKF backbone of this technique, however, scales badly with the number of features kept in the map state, and this meant that only around 10–15 features (matched using 11×11 pixel patches) could be tracked per frame; all but 3% of every image was ignored for the purposes of image alignment, and this sets a limit on the mosaicing quality which can be achieved.

Recently in real-time 3D camera tracking, methods based not on filtering but parallel pose estimation relative to keyframes and global optimisation have enabled large amounts of image correspondence information to be used in all frames. This approach was pioneered by Klein and Murray's Parallel Tracking and Mapping (PTAM) system [7] where hundreds of feature points are tracked per frame and built into a globally consistent 3D model of a workspace. Importantly, PTAM demonstrated that only tracking relative to the nearest keyframe is necessarily required to run at frame-rate to maintain live operation. The global optimisation component of PTAM (bundle adjustment of scene points and keyframes) runs in a parallel thread and repeats only as often as processing resources allow at a fraction of frame-rate.

This decoupling of local motion tracking from building a consistent global world model has become a dominant methodology in more generic SLAM research in robotics, since the pioneering work of Lu and Milios [8] and the first full implementation of a sequential mapping algorithm combining local tracking with interleaved global optimisation by Gutmann and Konolige [9], in this case with 2D laser scan data. With this interleaved approach, one is free to choose raw data alignment methods for the local tracking component, and the SLAM 'map' consists of the historically estimated sensor poses rather than feature locations.

In our work, we adapt this parallel tracking/optimisation approach to live video mosaicing, and make use of a state of the art whole image alignment method both for local rotation tracking and at the heart of a parallel optimisation thread for globally consistent alignment of a set of keyframes spanning the whole viewsphere. We are also able to refine estimates of camera intrinsic parameters in this global optimisation. Whole image alignment, as opposed to feature tracking, densely makes use of all of the texture in the images to permit registration which is as accurate as possible. Further, we show that a hierarchical implementation via an image pyramid permits the tracking to be efficient while maintaining a wide basin of convergence allowing very rapid camera rotation to be tracked.

Still one of the most widely used methods for estimating the warp between images, the Lucas-Kanade [10] method is based on the iterative minimisation of a cost function related to how well one reference image matches that of a warped comparison image. The parameters of the warp define the dimensionality of this space. By computing the derivative of the cost function with respect to the warp parameters, the parameter space gradient can be 'surfed' to a minimum, which may or may not be the global minimum.

Within our system, we make extensive use of the technique proposed by Malis, named Efficient Second-order Minimisation (ESM) [11] which instead finds the second order minimiser of the cost function while using only first order terms. This provides stable convergence in fewer iterations than the Lucas-Kanade method.

2 Method

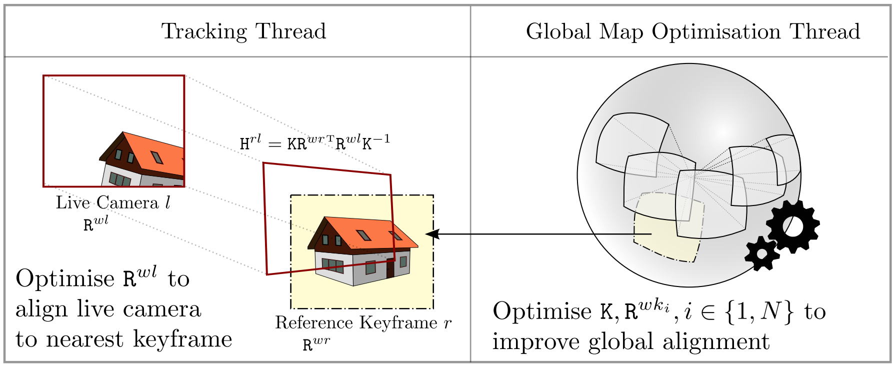

Tracking Thread
Global Map Optimisation Thread
$\mathtt{H}^{rl} = \mathtt{K}\mathtt{R}^{wr\top}\mathtt{R}^{wl}\mathtt{K}^{-1}$
Live Camera l
$\mathtt{R}^{wl}$
Optimise $\mathtt{R}^{wl}$ to align live camera to nearest keyframe
Reference Keyframe r
$\mathtt{R}^{wr}$
Optimise $\mathtt{K}, \mathtt{R}^{wk_i}, i \in \{1, N\}$ to improve global alignment

Fig. 1. System overview showing separation of tracking and mapping.

Our algorithm is split into two tasks which run as parallel threads on a multi-core PC: a) tracking from a known map, and b) global map maintenance and optimisation (see Figure 1), an approach inspired by PTAM [7]. In the first 'tracking' thread, we use the direct, whole image second order optimisation method ESM of Malis [11], with further contributions from Mei *et al.* [12], which we implement on graphics hardware for high-quality real-time tracking relative to our map. In the second parallel thread, we run a global optimisation procedure also based on ESM which adjusts the estimated orientations of all keyframes of our map and camera intrinsics simultaneously. This allows us to produce globally consistent mosaics in real-time. We remove radial distortion from all live frames as they enter our system, and deal only with perspective images from then on. We use a third party tool to establish the distortion parameters. Additionally, we descrive an automatic method for relocalisation if tracking should fail, allowing the current mosaic to be re-joined without corruption.

Keyframe Map Within our system, we store a collection of key historic camera poses with associated image data, which we call keyframes. Keyframes within our map are related to one another by a 3DOF rotation. We store the current estimate of a keyframe's pose as a rotation matrix $\mathtt{R}^{wk}$ relating the camera's local frame of reference, k, to that of the world, w.

Tracking When tracking commences, we set the first live image to be our first keyframe, k_0 with pose $\mathtt{R}^{wk_0}$ set to the identity. For each subsequent live frame, we use the previous live pose to select the closest keyframe from our map. We estimate the current pose by considering the image warp between this keyframe and the current image, which in turn allows us to estimate the relative motion.

Exploration As tracking continues, we create new keyframes and add them to the map if the overlap between our current image and closest keyframe becomes too small and falls below a threshold. Keyframes which we add inherit the pose of the live camera at that time.

2.1 Local Motion Estimation

For local motion estimation, we update our current pose estimate, $\mathtt{R}^{wc}$, by considering the live image and a reference keyframe r with known pose, $\mathtt{R}^{wr}$.

For two cameras in a general configuration observing a plane, we can describe pixel correspondence within their images by a plane induced homography. Cameras which purely rotate, however, allow us to disregard the scene entirely. Defining $\mathtt{H}^{ba}$ as the homography that transfers points imaged in camera a to the equivalent points in camera b, we can write $\mathtt{H}^{ba}$ as a function of $\mathtt{R}^{ba}$:

$$\mathtt{H}^{ba} = \mathtt{K}\mathtt{R}^{ba}\mathtt{K}^{-1} \ , \tag{1}$$

where $\mathtt{K}$ is the 3×3 camera intrinsic calibration matrix:

$$\mathtt{K} = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} . \tag{2}$$

This enables us to generate views from rotated 'virtual' cameras by warping an existing image. Our frame to frame tracking problem is then to find an update to the parameters of the plane induced homography $\mathtt{H}^{lr}$ which best reflects the warp between reference keyframe r and our live camera l.

Following the method of Malis [11], we parametrise updates to our pose using the Lie Algebra. The class of 3×3 rotation matrices belong to the Lie Special Orthogonal group SO(3). This group can be minimally parametrised around the identity by a three-vector belonging to the associated Lie Algebra $\mathfrak{so}(3)$. This parametrisation is locally Euclidean about $\mathbf{0}$, which is important for the ESM method. An element $x \in \mathfrak{so}(3)$ is related to a member $R(x) \in SO(3)$ through the matrix exponential map, where elements of x form coefficients for the group generators, $A_i, i \in [1, 2, 3]$:

$$\mathtt{R}(x) = \exp\left(\sum_{i=1}^{3} x_i \mathtt{A}_i\right). \tag{3}$$

Given a current estimate of the rotation, $\hat{\mathtt{R}}^{lr}$, and an update parametrised by $x \in \mathfrak{so}(3)$, $\mathtt{R}^{lr}(x)$, we update our estimate using the following rule:

$$\hat{\mathtt{R}}^{lr} \longleftarrow \hat{\mathtt{R}}^{lr}\mathtt{R}^{lr}(x). \tag{4}$$

We can now define an objective function describing the sum of squared differences between pixels in the live and reference images related by the homography, itself a function of the current rotation estimate $\hat{\mathtt{R}}^{lr}$, and the update x:

$$f(x) = \frac{1}{2} \sum_{p_r \in \Omega_r} \left[\mathcal{I}^l \left(\mathtt{H}\left(\hat{\mathtt{R}}^{lr}\mathtt{R}(x)^{lr}\right) p_r \right) - \mathcal{I}^r\left(p_r\right) \right]^2. \tag{5}$$

$\mathcal{I}^r$ and $\mathcal{I}^l$ represent the reference keyframe and live image respectively. The sum is formed from each pixel p_r in the set of pixels Ω_r defined in the reference image.

It can be shown that, up to second order, this function is minimised at x_0 (Equation 6), where $+$ is the pseudo-inverse and J the Jacobian relating change in parameters to changes in the cost function (Equation 7) [12]:

$$x_0 = -J^{+} f(0) \tag{6}$$

$$J = \left(\frac{J_{\mathcal{I}^l} + J_{\mathcal{I}^r}}{2}\right) J_w J_{\mathtt{K}} J_{\mathtt{R}} J_x. \tag{7}$$

The reader is asked to refer to [11–13] for details, including the definition of these Jacobians. The special formulation of these Jacobians taken about the reference and current images and the subsequent minimisation of this objective function is what is referred to as Efficient Second-order Minimisation (ESM).

If we instead write $f(x)$ explicitly as the norm of a residual difference vector $\mathbf{d}$ (Equation 8), where each row corresponds to a pixel in Ω_r (Equation 9), we see that the size of the system can be reduced by solving instead its normal equations (Equation 10):

$$f(x) = \frac{1}{2} \|d(x)\|^2 \tag{8}$$

$$d_{p_r}(x) = \mathcal{I}^l \left(\mathtt{H}\left(\hat{\mathtt{R}}^{lr}\mathtt{R}^{lr}(x)\right) p_r \right) - \mathcal{I}^r\left(p_r\right) \tag{9}$$

$$x_0 = -(J^{\mathrm{T}} J)^{-1} J^{\mathrm{T}} f(0). \tag{10}$$

Since J has dimensions *num pixels* $\times 3$, $J^{\mathrm{T}} J$ (a 3×3 matrix) is significantly smaller than J, and can be computed by summing the individual outer products of rows of J. We progress by iteratively solving this non-linear least squares system, applying the update $\hat{\mathtt{R}}^{lr} = \hat{\mathtt{R}}^{lr}\mathtt{R}^{lr}(x_0)$ until convergence.

Upon convergence, $\hat{\mathtt{R}}^{lr}$ represents the transformation between the live and reference cameras. Applying this to consecutive frames from a video sequence could form the basis for a visual odometry system. Here, instead, we match the current live image against the 'closest' keyframe in our map.

2.2 Global Map Optimisation

Joint global optimisation of all keyframes of the map and camera intrinsics occurs concurrently in a separate thread. We apply the ESM method to a more general objective function. We parametrise updates to pose through the Lie Algebra as before, but formulate updates to the camera intrinsic parameters by a vector, $k \in \mathbb{R}^4$, through exponentiation. Thus, $k = \mathbf{0}$ represents no change to the intrinsics. The update rule becomes:

$$\begin{pmatrix} f_u \\ f_v \\ u_0 \\ v_0 \end{pmatrix} \longleftarrow \begin{pmatrix} f_u e^{k_0} \\ f_v e^{k_1} \\ u_0 e^{k_2} \\ v_0 e^{k_3} \end{pmatrix} . \tag{11}$$

For N keyframes, our update vector x can be decomposed into rotation parameters, $r_i \in \mathfrak{so}(3)$, and intrinsic parameters: $x = (k, r_1, r_2, ...r_N)$. The objective function which we now wish to minimise includes all pairs of overlapping images:

$$f(x) = \frac{1}{2} \sum_j \sum_i \sum_{p_j \in \Omega_j} \left[\mathcal{I}^i \Big(\mathtt{H}^{ij}(x) p_j \Big) - \mathcal{I}^j \left(p_j \right) \right]^2 . \tag{12}$$

$$\mathtt{H}^{ij}(x) = \hat{\mathtt{K}}\mathtt{K}(k)\hat{\mathtt{R}}^{ij}\mathtt{R}^{ij}(r_i, r_j)(\hat{\mathtt{K}}\mathtt{K}(k))^{-1} \tag{13}$$

$$\hat{\mathtt{R}}^{ij}\mathtt{R}^{ij}(r_i, r_j) = (\hat{\mathtt{R}}^{wi}\mathtt{R}^{wi}(r_i))^{\mathrm{T}}\hat{\mathtt{R}}^{wj}\mathtt{R}^{wj}(r_j). \tag{14}$$

We calculate the incremental minimiser of this function x_0 using exactly the same machinery as before. Iterations of this minimisation take place continuously, helping to improve the map consistency.

Auto-calibration of camera intrinsics is particularly well posed in the case of a camera which only rotates [14]. In our system, the expected performance of calibration refinement is much further enhanced by our ability to match images automatically around full 360° panoramas, giving the potential for accurate calibration even for cameras with a narrow field of view.

2.3 Recovery from Tracking Loss

We have provided our SLAM system with a straightforward relocalisation capability similar in spirit to the 'small blurry image' method of PTAM [7] but which directly takes advantage of the main ESM pose estimate algorithm. If the camera becomes ‘lost’ then we aim to recover a pose estimate by simply attempting ESM pose estimation from a number of seed locations visible in our current mosaic, starting at the smallest image size in an image pyramid. Of the estimated warp parameters obtained, we refine the most photo-consistent estimate by performing more ESM iterations at higher resolutions in the pyramid. We use the poses of our keyframes as seed locations, but indeed any regular sample would be equally valid.

Computation time for relocalisation is proportional to the number of seed locations. For spherical mosaics, relocalisation need not be costly. When lost (measured using observed photoconsistency between the current keyframe and live camera), we run the relocalisation procedure on one in ten frames. This method operates well in environments with low perceptual aliasing.

3 Implementation

To achieve real-time performance, we make extensive use of commodity graphics hardware and the parallelism that this can afford. Graphics cards usually have a number of very simple, high throughput shaders that are ideal for stream processing tasks; taking quantities of data which are largely independent of each other and transforming this data in some way.

We use the portable graphics language Cg, which can run on the majority of today’s PCs and laptops. In this section, we will outline some of the more interesting implementation details of our system.

3.1 Real-Time Hierarchical ESM for Local Tracking

Our local tracking ESM implementation is split into three very simple stages targeting the graphics card, described below.

Hierarchical Construction After a frame is received from the video camera, it is uploaded as a texture on the GPU. Once in graphics memory, a fragment shader is invoked once for each desired level in a power-of-two reduction pyramid.

The fragment shader, which operates per pixel, simply takes the value of the average of the corresponding 4-block from the level above. This gets rendered back into a different texture of half the size. Typically, we use five levels in our pyramid which correspond to four invocations of this fragment shader. The individual levels of the pyramid are left on the graphics card and never downloaded to the CPU.

By first estimating the warp parameters between images at the smallest resolution in the pyramid, we benefit from a wider parameter-space convergence basin and lower processing costs. By assuming that per-pixel derivatives are meaningful at each of the levels, we are able to reuse our estimated warp parameters in the next highest resolution image and repeat. We can tune for performance/accuracy by setting how many iterations to perform at various levels.

Construction of Least Squares System Elements For every step in the ESM method, Jacobian terms common to all pixels are computed on the CPU (J_K, J_R, J_x). This leaves the data-centric terms (J_w, $J_{\mathcal{I}^l}$, $J_{\mathcal{I}^r}$) to be computed on the GPU. $J_{\mathcal{I}^l}$ and $J_{\mathcal{I}^r}$ are computed by central difference. The 9×3 matrix $J_K J_R J_x$ is loaded onto the GPU as parameters to a fragment shader in three 3×3 blocks, which are supported as primitives in the Cg language.

Invoking the fragment shader runs a simple Cg function per pixel p^r that enables us to compute the appropriate row of J, J_{p^r} and the residual d_{p^r}. This shader function also computes the outer product $J_{p^r}^{\mathrm{T}} J_{p^r}$ and product $J_{p^r}^{\mathrm{T}} d_{p^r}$. Since $J_{p^r}^{\mathrm{T}} J_{p^r}$ is symmetric, it has 6 unique elements; $J_{p^r}^{\mathrm{T}} d_{p^r}$ has 3. The shader function returns these 9 values as pixel 'colours' across three floating point RGBA textures stored on the GPU. We use OpenGL framebuffers to enable this.

Reduction to Linear System Given our three textures, where a channel of each image, for every pixel p^r, corresponds to elements of $J_{p^r}^{\mathrm{T}} J_{p^r}$ and $J_{p^r}^{\mathrm{T}} d_{p^r}$, we wish to compute $J^{\mathrm{T}} J$ and $J^{\mathrm{T}} d$. This involves summing the channels of each pixel, which we perform in two stages. The first is a vertical reduction in another Cg fragment shader. This shader is invoked on an output set of images containing a single row. For each pixel, this shader sums the pixels of the input images in the same column.

Finally, we download these three row images to the CPU, where the final horizontal reduction takes place to a single vector, which is unpacked into the appropriate matrix and vector. Here, it is solved using an efficient Cholesky decomposition.

3.2 Rendering

Two common approaches to visualising rotational mosaics are spherical and cylindrical projection. A spherical mosaic is visualised from within the center of a view-sphere, where images are projected to the sphere surface. Cylindrical projections are instead projected on to a cylinder, which we can then unwrap into a single image, visualising all of the mosaic at once.

We again make use of Cg shaders to enable us to visualise the full quality, blended mosaic live, and for correctly sampling from the constituent keyframes.

Spherical Panorama For rendering a spherical panorama, we treat our virtual (OpenGL) camera much like a keyframe, positioned at the origin and parametrised by the camera to world transform $\mathtt{R}^{wc}$. We can map image space coordinates from our OpenGL viewport to a keyframe k by composing the homography $\mathtt{H}^{kc} = \mathtt{K}\mathtt{R}^{wk^{\mathrm{T}}}\mathtt{R}^{wc}\mathtt{K}^{-1}$.

We use a shader which we invoke once for each keyframe within the field of view of the virtual camera, passing in as a parameter the homography $\mathtt{H}^{kc}$ which enables us to place the keyframe within the viewport. This shader, operating per-pixel, simply adds the keyframe's colour value to the colour already in the frame buffer associated with the viewport. Additionally, it adds 1.0 to the alpha channel for the pixel which serves as a counter.

Finally, we invoke another normalisation shader, which simply divides the Red, Green and Blue channels by the alpha channel. The result is a panorama where each keyframe is displayed blended with equal weight. One of the nice aspects of this method is that image fusion occurs in the space of the viewport. This means that each keyframe, whose pixel data is not sampled to the same 'grid' in viewport space, gets mixed to form an image of higher resolution of the constituent images. Dependent on the quality of image registration, this can enable 'super resolution' images to be displayed at frame rate.

Cylindrical Panorama To create cylindrical panoramas, we use similar machinery as for spherical panoramas. Within the shader, the u and v viewport coordinates are interpreted as yaw (ψ) and pitch (θ) in the range $[-\pi, +\pi]$ and $[-\frac{\pi}{2}, +\frac{\pi}{2}]$ respectively.

For each keyframe, we invoke the shader, where, for each pixel we then compute the desired image ray described by the unit vector $\hat{\mathbf{r}}$,

$$\hat{\mathbf{r}} = \left(\cos\theta\cos\psi, \sin\theta, \cos\theta\sin\psi\right)^{\mathrm{T}} . \tag{15}$$

This is transferred into the frame of reference of the keyframe using the virtual camera to keyframe rotation matrix, $\mathtt{R}^{kc}$, which is uploaded as a parameter to the shader. Finally, the camera intrinsic matrix can be used to map this to keyframe image-space coordinates. Given this correspondence, we proceed as with the spherical panorama.

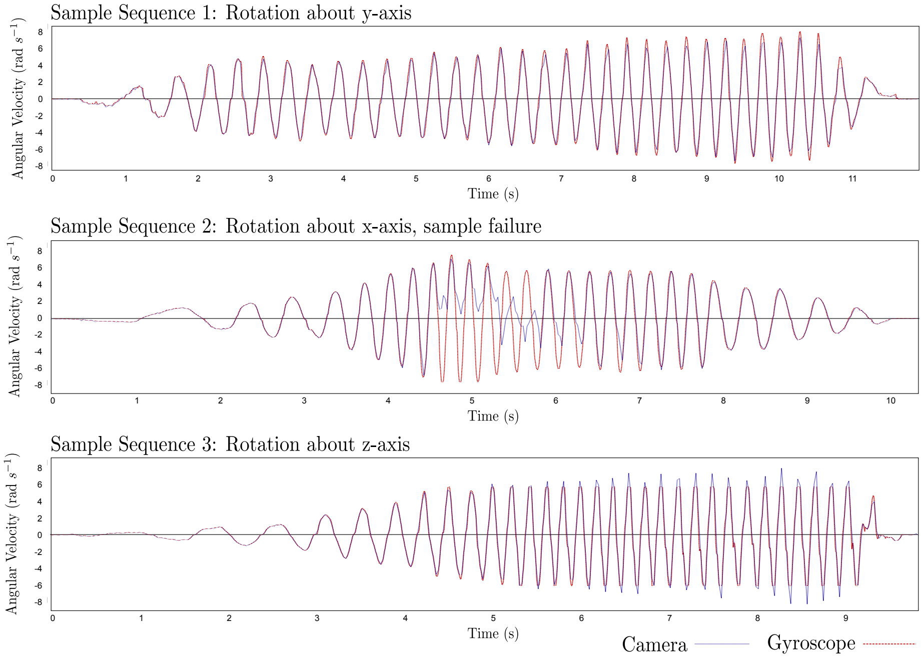
4 Results

We wish to evaluate our system against two criteria; how accurately local motion is estimated, and how consistently frames are registered into a final mosaic.

In all of the results, as our submitted video also highlights, mosaics were computed incrementally and rendered live at frame rate, a solid 30fps. We cap per frame ESM iterations to 48 at the 5^{th} level of the pyramid, 16 at the 4^{th}, 8 at the 3^{rd}, 4 at the 2^{nd}, and 2 at the 1^{st}. We use any remaining time to perform iterations at the 0^{th} level which corresponds to the original image — this typically is one, two or three iterations. We drop new keyframes when less than 80% of the current keyframe is visible.

4.1 Local Motion Estimation and Dynamics

To test the ability of our method to track dynamic local motion, we have compared the angular velocity output of our method against a solid state gyroscope bolted to the back of the camera, which was mounted on a tripod and oscillated to produce increasingly rapid motion (up to around 5 cycles per second) about each of its axes in turn.


Sample Sequence 1: Rotation about y-axis

Sample Sequence 2: Rotation about x-axis, sample failure

Sample Sequence 3: Rotation about z-axis

Fig. 2. Graphs illustrating high dynamic tracking performance; the plots show angular velocity estimates from our vision system compared with the output from a gyroscope as the camera was vigorously oscillated about each of the three camera-oriented axes in turn (y axis pan; x elevation, z cyclotorsion).

The characteristics of estimation are somewhat different depending on the axis of rotation, as the plots of Figure 2 illustrate. Angular velocity about the z-axis (cyclotorsion) is estimated very accurately. Note that the truncated peaks of the gyroscope data show that the tracking limits of the device were exceeded while visual tracking still continued accurately — our system was able to maintain fidelity about this axis in excess of 7 $rads^{-1}$, which is significantly faster than a camera would normally move in a tracking scenario.

Angular velocity about the y-axis, corresponding to camera pan, tracks the gyroscope data closely, with a very slight systematic under-estimation. We suspect that camera calibration may be the predominant cause, or a slight misalignment between the camera and gyroscope frames of reference.

The plot showing rotation about the x-axis, corresponding to camera elevation, demonstrates a failure case of visual tracking caused by extreme motion. The tracking under-shoots, and takes several oscillations to re-acquire correspondence with the keyframe against which it is tracking. If the motion was non-cyclic, it would be harder for the system to recover to an orientation fixed in the global frame without resorting to relocalisation. The system is least stable about this axis. We suggest that this is due to the narrower vertical field of view.

4.2 Global Consistency and Intrinsics Refinement

For evaluation of global registration, we present several cylindrically projected 360° panoramas (Figures 3, 5) captured with two different cameras, and with two different lenses for each camera. They are constructed by blending every keyframe of the map with equal weight, as described in Section 3.2, enabling us to visualise the quality of their alignment.

For areas of the mosaic formed from multiple images, pixel noise is significantly reduced, and the mosaic appears smoother. The different sampling pattern of keyframes and sub-pixel accuracy we achieve in alignment combine to create a super-sampling, or 'super-resolution' image, efficiently rendered in real-time on the graphics card.

Figure 4 demonstrates the importance of our joint estimation of camera intrinsic parameters, even for pre-calibrated cameras. Starting with intrinsics estimated from a third party camera calibration tool, and continuing with no intrinsics optimisation, the first mosaic in this figure appears fuzzy. Upon inspection we can see that the estimated loop length is longer than the actual length (in pixels), causing the images to bunch up (the enlargement of the whiteboard helps to convey this point). This is caused by intrinsic parameters which are wider than the actual camera. The second mosaic in this figure is the result of allowing our algorithm to optimise intrinsics as well as pose parameters (from the starting point of the first mosaic).

The mosaics in Figure 3 were generated from three different lenses, all at 640×480 resolution, and initialised with 'Generic' intrinsic calibration (nearest 10° FOV and central principal point). Table 4.2 shows the initial horizontal field of view, which was based on our knowledge of the lens, and the converged field of view estimate after a full loop was completed for these sequences.

Table 1. Calibration Refinement results for Different Cameras and Lenses. Calibration initialised from Quoted Horizontal Field of View (FOV), and refined by mosaicing cylindrical loops from 640×480 indoor sequences.

Camera	Lens	Lens Quality	Initial FOV Estimate	Refined FOV
Point Grey Flea2	Wide Angle	Good	70°	69.42°
Point Grey Flea2	TV Lens	Fair	50°	51.43°
Unibrain	Standard	Poor	50°	45.56°

4.3 Convergence to Global Minimum

The results from mosaicing based on poor initial intrinsics (Figure 3) help to motivate that our system has useful convergence properties. By including intrinsics in our optimisation, we help to enable loop closure by increasing the accuracy of our pose estimate when we come to complete a loop. By completing a loop

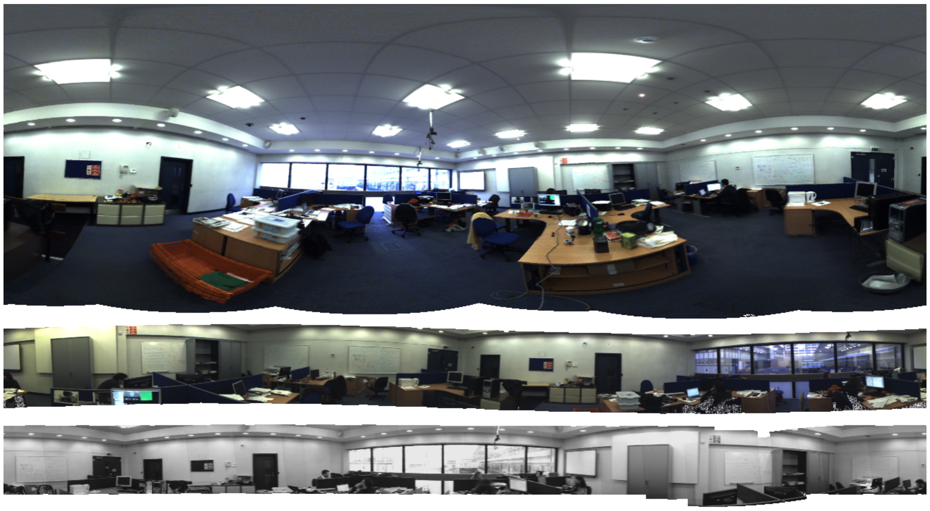


Fig. 3. 360° cylindrically-projected panoramas for three indoor sequences, taken with different lenses. Point Grey Flea2, 70° FOV wide angle (top, close to full sphere including full hemispherical upward coverage, 27 keyframes), 50° FOV TV Lens (middle, single horizontal loop trajectory, 17 keyframes), and Unibrain 45° FOV Standard lens (bottom, single horizontal loop trajectory, 19 keyframes).

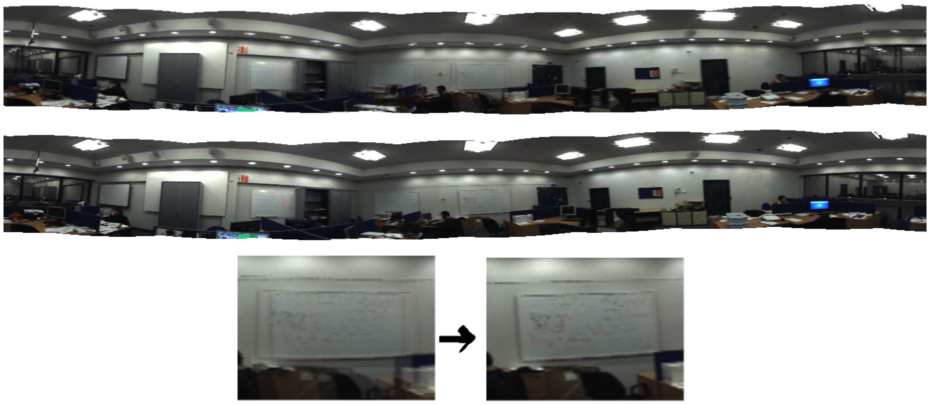


Fig. 4. Mosaicing with fixed intrinsics estimated from a third party calibration tool (top), compared against enabling live intrinsics estimation (middle). An enlargement of the whiteboard from the two mosaics, emphasising improvement in alignment, is shown at the bottom. The whiteboard is representative of several areas of the mosaic.

too soon, or too early, we are more likely to fall into local minima — especially if perceptual aliasing in this area is high.

Figure 5 shows an outdoor mosaic generated from rapid hand-held motion of a Unibrain webcam with a wide angle lens. Note that in this experiment the pure rotation assumption was approximately satisfied without a tripod due to the large distance to the scene. This scene contains high perceptual aliasing in the windows and building pillars, making loop closure difficult. For this sequence, we were unable to converge to a globally consistent mosaic from our generic 80° FOV calibration parameters. Instead, we started from the parameters estimated from a third party calibration tool.

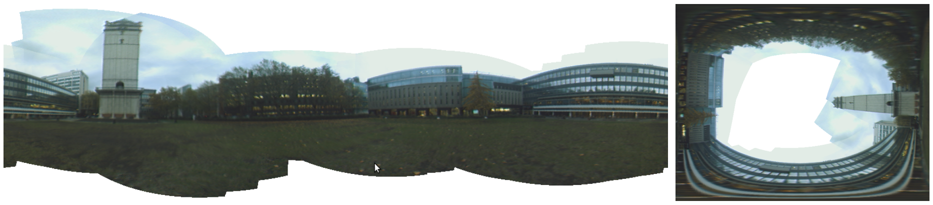


Fig. 5. 360° Tower panorama from 21 keyframes (live hand held Unibrain webcam, 320×240 resolution), shown in horizontally and vertically-oriented cylindrical projection. Note the vertical hole due to poor texture and cloud movement in the sky.

Time to convergence is another important evaluation criterion. Each iteration in our global minimisation is costly — forming the linear system from image data dominates computational time. Actually solving this system is cheap since spherical mosaics require only a relatively small number of keyframes. For this reason, computation time scales linearly with the number of pairs of overlapping pixels. For N keyframes, depending on keyframe alignment, this has a worst case complexity of $\mathrm{O}\left(N^2\right)$. In practice, our system achieves convergence within time in the order of seconds of completing a loop; often less than one second when a wide angle lens means that the number of keyframes to span a loop is low.

5 Conclusions

We have presented an algorithm based on full image alignment which produces accurate, globally consistent mosaics in real-time. Our key contribution is to show how state of the art image alignment can be used in a robust and accurate real-time mosaicing system which combines the best of a visual gyroscope, with its ability to track rapid motion, with the properties of a visual compass, able to function without long-term drift. We also demonstrate convincing automatic camera calibration refinement, and explain how real-time tracking and rendering can be comfortably achieved using commodity graphics hardware.

The clear extension to our method which we plan to investigate is the capability to track general motion viewing multi-planar scenes, such as building façades and room interiors. We can enforce strong priors in such environments and hope to demonstrate very fast, robust tracking and coarse model construction.

Acknowledgements

This research was supported by an EPSRC DTA studentship to S. Lovegrove and European Research Council Starting Grant 210346. We are very grateful to Richard Newcombe and other colleagues at Imperial College London for many useful discussions.

References

1. Szeliski, R.: Image alignment and stitching: A tutorial. Foundations and Trends in Computer Graphics and Vision **2** (2006) 1–104
2. Brown, M., Lowe, D.G.: Recognising panoramas. In: Proceedings of the International Conference on Computer Vision (ICCV). (2003)
3. Steedly, D., Pal, C., Szeliski, R.: Efficiently stitching large panoramas from video. In: Proceedings of the International Conference on Computer Vision (ICCV). (2005)
4. Szeliski, R., Shum, H.Y.: Creating full view panoramic image mosaics and environment maps. In: ACM Transactions on Graphics (SIGGRAPH). (1997)
5. Morimoto, C., Chellappa, R.: Fast 3D stabilization and mosaic construction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (1997)
6. Civera, J., Davison, A.J., Magallón, J.A., Montiel, J.M.M.: Drift-free real-time mosaicing. International Journal of Computer Vision (IJCV) **81** (2009) 128–137
7. Klein, G., Murray, D.W.: Parallel tracking and mapping for small AR workspaces. In: Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR). (2007)
8. Lu, F., Milios, E.: Globally consistent range scan alignment for environment mapping. Autonomous Robots **4** (1997) 333–349
9. Gutmann, J.S., Konolige, K.: Incremental mapping of large cyclic environments. In: International Symposium on Computational Intelligence in Robotics and Automation (CIRA). (1999)
10. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI). (1981)
11. Malis, E.: Improving vision-based control using efficient second-order minimization techniques. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2004)
12. Mei, C., Benhimane, S., Malis, E., Rives, P.: Efficient homography-based tracking and 3-D reconstruction for single-viewpoint sensors. IEEE Transactions on Robotics (T-RO) **24** (2008) 1352–1364
13. Silveira, G., Malis, E., Rives, P.: An efficient direct approach to visual SLAM. IEEE Transactions on Robotics (T-RO) **24** (2008) 969–979
14. Agapito, L., Hayman, E., Reid, I.: Self-calibration of rotating and zooming cameras. International Journal of Computer Vision (IJCV) **45** (2001) 107–127