Improving the Agility of Keyframe-Based SLAM

Georg Klein and David Murray

Active Vision Laboratory, University of Oxford, UK
{gk,dwm}@robots.ox.ac.uk

Abstract. The ability to localise a camera moving in a previously unknown environment is desirable for a wide range of applications. In computer vision this problem is studied as monocular SLAM. Recent years have seen improvements to the usability and scalability of monocular SLAM systems to the point that they may soon find uses outside of laboratory conditions. However, the robustness of these systems to rapid camera motions (we refer to this quality as agility) still lags behind that of tracking systems which use known object models. In this paper we attempt to remedy this. We present two approaches to improving the agility of a keyframe-based SLAM system: Firstly, we add edge features to the map and exploit their resilience to motion blur to improve tracking under fast motion. Secondly, we implement a very simple inter-frame rotation estimator to aid tracking when the camera is rapidly panning – and demonstrate that this method also enables a trivially simple yet effective relocalisation method. Results show that a SLAM system combining points, edge features and motion initialisation allows highly agile tracking at a moderate increase in processing time.

1 Introduction

Real-time camera tracking or ego-motion recovery in known scenes is a well-studied problem in computer vision with immediate applications in robotics and augmented reality. Recent years have also seen an interest in the problem of tracking in unknown or only partially known scenes. This is a more difficult problem because the computer must not only estimate camera pose but also calculate a map of its environment. The prevalent approach is that of SLAM (Simultaneous Localisation and Mapping) which differs from recursive Structure-from-Motion (SfM) in that greater care is taken properly to correlate the emergent structure and camera pose.

A real-time, monocular SLAM implementation was demonstrated by Davison in 2003 ([1] is an up-to-date version). Davison's implementation is based on an Extended Kalman Filter which is updated every frame at $O(N^2)$ cost relative to map size, and this mandates the use of sparse and relatively small maps; further, tracking is based on small texture patches which are destroyed by motion blur; finally, the fact that every frame's measurements are integrated into the map requires a conservative approach to data association (i.e. small modelled accelerations and search ellipses). Consequently and subsequently there have been attempts to improve both the scalability and robustness of monocular SLAM. Chekhlov et al [2] improve the reliability of data association by employing the SIFT [3] descriptor, which allows for far bigger search ellipses and hence more agile tracking. Williams et al [4] also attach descriptors to map points, but do not

D. Forsyth, P. Torr, and A. Zisserman (Eds.): ECCV 2008, Part II, LNCS 5303, pp. 802–815, 2008.

use these during tracking, but instead to re-localise the camera in the event of a tracking failure. Eade and Drummond [5] employ a different statistical framework which allows denser maps, which improves tracking quality. Here, we start with our previous work [6] which de-couples the map-making process from frame-to-frame tracking, performing classical batch structure-from-motion estimation on a small number of *keyframes* as one process while tracking the camera relative to the map as another. This approach allows for more aggressive tracking, since data association errors are not automatically integrated into the map, and it is briefly described in Section 3.

Despite the advances made in monocular SLAM, we still find that rapid camera motions are generally not tracked. While the emergence of relocalisation methods has removed the need completely to re-start a SLAM system every time the camera is moved too quickly, it would be preferable if tracking did not fail in the first place. The problem with tracking rapid camera motions is that it produces large and unpredictable motion in the captured image: this large motion makes it difficult reliably to track individual features (data association) and it also produces motion blur. We use small commodity cameras with integration times approaching 30ms in indoor scenes, and so the extent of motion blur in the image can be very large (at times over 100 pixels). Most current monocular SLAM implementations use some form of interest point detector to achieve frame-rate operation (often Rosten and Drummond's FAST detector [7]), but even small amounts of motion blur will wipe the image clean of corner responses. Even if the interest point detector did fire, substantial motion blur will render useless the small texture templates typically used for feature matching. The problem can be somewhat alleviated by using features at multiple pyramid levels (as done in [6]) but high-level features are relatively sparse and even these are unreliable with heavy blur: Ultimately, tracking texture patches in the presence of blur is difficult.

In contrast to corner points, edges in the image are more resilient to motion blur. For a start, they are one-dimensional, and so locally is motion blur: One would therefore expect that even in a heavily blurred image, some edge features (those parallel to the local direction of blur) may remain intact. Further, even edges which are affected by blur can be tracked: Klein and Drummond [8] demonstrated that blurred edges can be used for tracking in real-time if an estimate for the magnitude of motion blur is known a priori. This motivates us to include intensity edges into the SLAM map. Monocular SLAM with edges has already been demonstrated by Smith et al [9] and Eade and Drummond [10], but neither of these systems attempted to exploit edges for agile tracking, as we do here. We adopt Eade and Drummond's concept of an *edgelet* – a very short, locally straight segment of what may be a longer, possibly curved, line – as our representation of intensity edges in the world. Section 4 describes the edgelet representation and our method for adding and optimising edgelets to our keyframe-based SLAM implementation, and Section 5 describes how these edgelets can be tracked robustly.

While one might expect the addition of edges to a SLAM system to improve tracking performance at high camera velocities, the same is not true for high acceleration. Rapid unmodelled accelerations cause problems for data association and can further lead to the use of an incorrect blur estimate for edge tracking – for example, when the camera rotates suddenly from rest. In [8] Klein and Drummond employed inertial sensors to provide a rotation prediction for each frame to work around this problem. Here we

propose an alternative: In Section 6 we describe a procedure to estimate inter-frame rotation by full-frame direct minimisation. Finally, we show that this same method – together with the map's stored keyframes – allow a trivial relocalisation method to recover from tracking failure, and this is described in Section 7.

2 Related Work

A body of related work exists in the SfM literature for calibrated cameras. Viéville and Faugeras [11] developed an EKF that estimates the scene structure and the camera position using the measurements obtained from a system that tracked the positions of point and line features in the images. Faugeras et al [12] solved for camera motion and scene structure from three perspective images using an EKF to minimize an objective function based on the epipolar constraint. Related is Taylor and Kriegman's work [13] which proposed minimizing a cost function measuring the least squares distance between observed edge segments and the projection of reconstructed lines, and Bartoli and Sturm [14] discuss optimal line parametrisations for the the minimisation procedure. SfM using lines with uncalibrated cameras has also been demonstrated, leading to the exposition of Shashua [15] on projective tri-focal constraints.

As mentioned in the introduction, there have been previous demonstrations of monocular SLAM using edge features. The approaches of Smith et al [9] and Eade and Drummond [10] are very different, and this is in part due to differences in their underlying SLAM mechanisms: [9] employs an EKF which scales poorly with map size, and so this approach attempts to add large straight lines with two well-defined end-points to the map. By contrast [10] uses FastSLAM 2.0 and so can afford to insert large numbers of features into the map, and this is exploited by splitting large lines (straight or slightly curved) into small segments called edgelets, which are added to the map as independent units. Since the decoupled SLAM system used here scales easily to thousands of features we adopt a variant of the latter approach here.

3 Keyframe-Based SLAM Framework

This section provides a brief overview of the SLAM and maths framework [6] which is also used here. The SLAM system is split into two separate processing threads, which run simultaneously on a dual-core computer. One thread processes the 30Hz, 640×480 pixel, 8bpp grey-scale video input from an attached fire-i camera with a 2.1mm wide-angle lens: This is the tracking thread, which tracks the pose of the camera relative to the map, assuming that the map is fixed and certain. The second thread creates, expands and maintains the map from a small subset of these frames which are called keyframes. Features are added upon triangulation between two keyframes, and feature positions and keyframe poses are jointly optimised using bundle adjustment.

The map consists of a set of keyframes and 3D feature points. We define a world coordinate frame $\mathcal{W}$, a moving camera-centered coordinate frame $\mathcal{C}$, and each keyframe i also has an associated camera-centered coordinate frame $\mathcal{K}_i$. The jth point's position is stored as a 4-vector $\boldsymbol{p}_{j\mathcal{W}} = (\, x_{j\mathcal{W}} \; y_{j\mathcal{W}} \; z_{j\mathcal{W}} \; 1 \,)^T$ in the world coordinate frame. The ith keyframe stores its world-from-keyframe coordinate frame transformation $E_{\mathcal{W}\mathcal{K}_i}$, a

three-octave image pyramid from the full 640×480-pixel image down to a 80×60-pixel scaled version, and a record of all point features measured in this keyframe.

Coordinate frame transformations are members of the Lie group SE(3), the set of 3D rigid-body transformations, and are expressed as 4×4 matrices with a rotation and a translation component

$$E_{\mathcal{AB}} = \begin{bmatrix} R_{\mathcal{AB}} & \boldsymbol{t}_{\mathcal{AB}} \\ 0\,0\,0 & 1 \end{bmatrix}. \tag{1}$$

The subscript "$\mathcal{AB}$" is read as "coordinate frame $\mathcal{A}$ from coordinate frame $\mathcal{B}$", such that $E_{\mathcal{AC}} = E_{\mathcal{AB}}E_{\mathcal{BC}}$. Changes to coordinate frame transformations are achieved via another (small) transformation denoted M and minimally parametrised by a six-vector $\boldsymbol{\mu}$ via the exponential map

$$E'_{\mathcal{AB}} = ME_{\mathcal{AB}} = \exp(\boldsymbol{\mu})E_{\mathcal{AB}}, \tag{2}$$

where in this case the first and last three elements of $\boldsymbol{\mu}$ denote respectively translation and rotation about the x-, y-, and z- axes of coordinate frame $\mathcal{A}$. This representation allows for very easy differentiation of projection equations with respect to any change in coordinate frame transformations.

4 Adding Edgelets to Keyframe-Based SLAM

This section describes the method by which edgelet landmarks can be added to the SLAM system described above, so that both point feature and edgelets are tracked. We use [10]'s definition of an edgelet in an image: a locally straight, short segment of a large intensity discontinuity in the image.

4.1 Parametrisation

For each edgelet, two geometrical properties must be recorded: the position of the center of the edgelet and the direction of the edgelet. [10] store these five degrees of freedom in six numbers, using a position vector and a unit vector of direction. We find it convenient to overparametrise even further, storing each edgelet as a coordinate frame transformation $E_{\mathcal{WE}}$, where $\mathcal{E}$ is an edgelet-local coordinate frame. (With 12 internal numbers and 6 DOF this representation is far from minimal but it simplifies the 4-DOF differentiations needed later.) In the coordinate frame $\mathcal{E}$ the edgelet center is located at the origin and the edgelet direction is parallel to the z-axis, as illustrated in Fig. 1. To set the last degree of freedom, we further align the x-axis to point in the direction from dark to light in the frame in which it was first observed. While this information could be used to determine edgelet visibility, we ignore it in favour of the common method of inferring this direction from the edgelet direction, and recording visibility information in a different way (see Sect. 5.2).

4.2 Projecting and Finding Edgelets in Keyframes

To measure an edgelet in a keyframe, a good prior estimate of its position is first projected into the full-size image. The edgelet center is transformed to the keyframe's

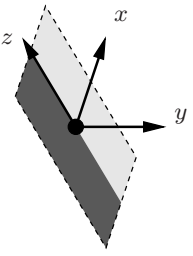

Fig. 1. The edgelet-centered 3D coordinate frame $\mathcal{E}$

camera-centered coordinate frame and then projected into pixel coordinates:

$$\boldsymbol{p}_{\mathcal{K}} = E_{\mathcal{K}\mathcal{E}} \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \tag{3}$$

$$\boldsymbol{c} = \begin{pmatrix} c_u \\ c_v \end{pmatrix} = \text{CamProj}(\boldsymbol{p}_{\mathcal{K}}) \tag{4}$$

where the CamProj$(\cdot)$ function performs perspective division by z followed by the application of a calibrated camera model as in [6], which uses the radial distortion model of [16]. The edgelet direction $\boldsymbol{d}$ is obtained by differentiating the above with respect to motion along the z-axis in the $\mathcal{E}$-frame:

$$\boldsymbol{d} = \frac{\partial}{\partial \mu_3} \text{CamProj} \left(E_{\mathcal{K}\mathcal{E}} M \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \right). \tag{5}$$

where μ_3 parametrises z-translation of the motion matrix M in the $\mathcal{E}$-frame. This is equivalent to projecting the rotated edgelet's z-axis into the image as done in [10].

Once projected, edgelets are measured directly in the full-size image, in a manner similar to most edge trackers [17,8,9]. The nearest locally-straight image edge is found by placing an edge of length 18 pixels in the image at the hypothesised position, populating five sample points along this edge, and then performing perpendicular searches at each sample point to find local gradient maxima of the correct polarity to sub-pixel accuracy. Data association is resolved by proximity (the perpendicular search range is short at only five pixels).

A straight line is fitted to the search results and the measurement accepted as valid only if the residual error is smaller than a tight threshold: this is to reject measurements of edgelets which are locally curved, or blurred, or confusingly close to other edges. The detected edgelet is parametrised as an infinite line $\boldsymbol{a} + \lambda \boldsymbol{b}$ as illustrated in Fig. 2.

4.3 Adding New Edgelets to the 3D Map

In both [9] and [10], new edges are added with initially unknown depth, and depth values calculated iteratively over a few frames of hopefully non-degenerate camera motion over which data association for the new edges must be maintained. This method is incompatible with keyframe-based SLAM for a number of reasons: 1. We do not want

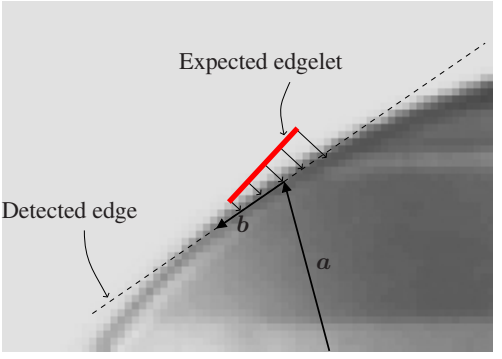

Fig. 2. Edgelet measurements in a keyframe

to impose the requirement that frame-to-frame motion should be smooth; 2. we want large quantities of features, and tracking large numbers of partially initialised features-in-making would place an unacceptably large computational burden on the frame-to-frame tracker; 3. it is not clear how to use the information from frame-to-frame depth estimation in subsequent bundle adjustments in a correct manner, short of just discarding it. By contrast, in [6] points are added by epipolar search and direct triangulation from two keyframes and we use an analogous approach here.

Every time a new keyframe is added to the map, a list of candidate edgelets which might be added to the map is generated. Where [10] suggest a fast edgelet extraction method which can be performed every frame while tracking, the map-making thread here is less bound to frame-rate constraints: We have time to perform a full Canny edge extraction [18] on the keyframe, with a modification to the edgel linker stage which breaks linked chains at points of high curvature. The result is a series of long, prominent, and reasonably straight edges in the image. (We also keep an image of all the maximal edgels, whether or not they were suppressed by the linker.) Candidate edgelets are formed by splitting these long edges into edgelet-length segments. Due to the use of Gaussian blur during the Canny extraction, edgel locations are not fully reliable and so the location of each candidates is refined by direct measurement in the keyframe as per Sect. 4.2, and candidates which are too close in the image to edgelets already existing in the map are rejected.

For each candidate edgelet, we perform an epipolar search in another near-by target keyframe to attempt to triangulate the edgelet. We do not search the entire epipolar line from zero to infinite depth, but rather restrict the search to a likely depth range based on the depth of features already found in the keyframe. The epipolar line is projected and rasterised into the target keyframe as a piecewise linear approximation (since the lenses used exhibit radial distortion, the epipolar line is not straight), and any maximal edgel encountered along the epipolar line is considered a potential match.

A single epipolar search can return a large number of matches in this way, and matches are culled by a number of heuristics. Often half of the matches can be rejected from light-to-dark polarity. Matches with gradients perpendicular to the epipolar line are further rejected. For any remaining match, an edgelet measurement in the target

keyframe is attempted as per Sect. 4.2, and this removes matches which are curved or too short. Any remaining match defines a viable 3D edgelet hypothesis: The source and target keyframes' edgelet measurements each define a plane in 3D space in which the edgelet must lie, and these are intersected to yield an infinite 3D line. The edgelet's position along this is defined by its center in the source keyframe. Each hypothesis is now checked by attempting a measurement in a third keyframe; a new edgelet is only added to the map if exactly one hypothesis is successfully tested against this keyframe.

For any two keyframes chosen as source and target keyframes, there exists with high likelihood a set of edgelets which cannot be successfully triangulated due to the aperture problem (for example if the edgelets are parallel to the camera translation between the two views). Such incompatibility is detected, and triangulation of these edgelets is attempted again with a different, more orthogonally translated, keyframe.

4.4 Optimising the Map Using Bundle Adjustment on Points and Lines

Edgelets are added to the map with at least three measurements – one each from the source, target, and check keyframes – and more measurements are attempted in all old keyframes and all keyframes subsequently added. Whenever information is added to the map, the position of all edgelets, points and keyframes is optimised through bundle adjustment. To incorporate edgelets into the bundle adjustment requires the definition of an objective function, and the derivation of its differentials with respect to both keyframe pose parameters and the parameters of the edgelets.

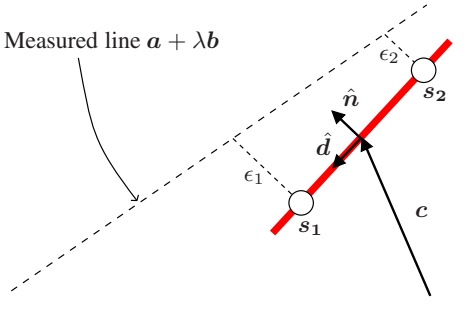

Fig. 3. Edgelet objective function measurements

Keyframe edgelet measurements provide two degrees of freedom, which can be expressed in a variety of ways. Here we use two perpendicular distances as illustrated in Fig. 3: Two virtual sample points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ with projections $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ are initialised at a 5-pixel distance from the projected edgelet center, and the perpendicular distances from these sample points to the edge detected in the keyframe forms the error metric. Compared to a distance-plus-angle metric, this has the advantage that errors are expressed in pixel units, which makes them compatible with the M-Estimator also used for point features. The sample points are symmetrically distributed in space about the edgelet center. Represented in the $\mathcal{E}$-frame they have coordinates

$$\boldsymbol{p}_{1\mathcal{E}} = \begin{pmatrix} 0 & 0 & \frac{-5.0}{|\boldsymbol{d}|} & 1 \end{pmatrix}^T, \quad \boldsymbol{p}_{2\mathcal{E}} = \begin{pmatrix} 0 & 0 & \frac{5.0}{|\boldsymbol{d}|} & 1 \end{pmatrix}^T \tag{6}$$

and their projections in the image are $\boldsymbol{s}_n = \text{CamProj}(E_{\mathcal{KE}}\boldsymbol{p}_{n\mathcal{E}})$. The error quantities ϵ are the edgelet-perpendicular distances from each sample point to the measured line

$$\epsilon_n = \hat{\boldsymbol{n}} \cdot \left(\boldsymbol{a} + \left(\frac{\hat{\boldsymbol{d}} \cdot \boldsymbol{s}_n - \hat{\boldsymbol{d}} \cdot \boldsymbol{a}}{\hat{\boldsymbol{d}} \cdot \hat{\boldsymbol{b}}} \right) \hat{\boldsymbol{b}} - \boldsymbol{s}_n \right) \tag{7}$$

where $\hat{\boldsymbol{n}}$ is perpendicular to $\boldsymbol{d}$ and a hat $\hat{\cdot}$ denotes a unit-normalized vector.

Differentials of the distance metric are obtained by considering the motion of points $\boldsymbol{p}_n$ in direction $\hat{\boldsymbol{n}}$ due to changes in keyframe pose and edgel parameters; for keyframe pose changes $\boldsymbol{\mu}_{\mathcal{K}}$ and edgelet motions $\boldsymbol{\mu}_{\mathcal{E}}$ they are

$$\frac{\partial \epsilon_n}{\partial \mu_{\mathcal{K}m}} = \hat{\boldsymbol{n}} \cdot \frac{\partial}{\partial \mu_{\mathcal{K}m}} \text{CamProj}\left(ME_{\mathcal{KE}}\boldsymbol{p}_{i\mathcal{E}}\right),\ m = 1..6 \tag{8}$$

$$\frac{\partial \epsilon_n}{\partial \mu_{\mathcal{E}m}} = \hat{\boldsymbol{n}} \cdot \frac{\partial}{\partial \mu_{\mathcal{E}m}} \text{CamProj}\left(E_{\mathcal{KE}}M\boldsymbol{p}_{i\mathcal{E}}\right),\ m = 1, 2, 4, 5. \tag{9}$$

Only four motion parameters are considered for edgelet motion: those that have no influence on the reprojection error (translation along and rotation about the z-axis) are ignored.

5 Frame-Rate Tracking with Edgelets

5.1 Edgelet Search

The method described in Sect. 4.2 to find edgelets in keyframes is based on the premise that keyframes are not significantly corrupted by motion blur and a very good prior pose estimate exists: This allows edgelets to be found with high precision and using a very short search range. Such guarantees do not exist for frame-to-frame camera tracking, and edgelet tracking should then be fast, have a large search range, and be tolerant of motion blur. Further, while 2-DOF edgelet measurements are useful to determine edgelet orientation when building the map, the second degree of freedom is only marginally useful for constraining camera pose: For this task, a single-DOF measurement is sufficient. We therefore use a different method for measuring edgelets in the frame-rate tracker than we use for making the map.

Each edgelet is measured by performing a single perpendicular search from the edgelet center, providing a perpendicular distance measure ϵ. To provide some resilience to motion blur, we adopt the search method of Klein and Drummond [8]: Instead of searching for an intensity step in the image, the motion blur expected for each edgelet is calculated from a motion model, and the image is searched for an intensity ramp of the same length as the expected blur. The ramp search is performed by extracting a 1D intensity signal sampled along the edge normal, and cross-correlating with a single cycle of a sawtooth wave of period l. l is the edge-normal component of expected motion blur and is calculated by simply multiplying the expected camera motion $\boldsymbol{\gamma}_{\mathcal{C}}$ during exposure by each edgelet's measurement Jacobian:

$$l = J\boldsymbol{\gamma}_{\mathcal{C}} \tag{10}$$

where J is the 1×6 Jacobian of an edgelet's perpendicular displacement with respect to changes in camera pose (with $\hat{\boldsymbol{n}}$ calculated as before)

$$J_m = \frac{\partial \epsilon}{\partial \mu_{\mathcal{C}m}} = \hat{\boldsymbol{n}} \cdot \frac{\partial}{\partial \mu_{\mathcal{C}m}} \text{CamProj}\left(ME_{\mathcal{CE}}\begin{pmatrix}0&0&0&1\end{pmatrix}^T\right),\ m = 1...6. \tag{11}$$

In contrast to Klein and Drummond [8] we search only for edge responses of the correct polarity. Further, in [8] each edgel search used the same maximum search length r_{search} and edge response threshold t_{min}. Here, we allow these parameters to vary with each edgelet, as described below.

5.2 Edgelet Visibility Table

Data association for edge tracking can be challenging because one intensity edge looks much like the next. Many systems [10,9] use only a 1-bit descriptor (the edge polarity) while [8] does not even use this. For SLAM systems without a pre-made CAD model to fall back on, the problem is exacerbated by the lack of prior occlusion information: The SLAM system will still look for an edge which has become hidden behind another, and likely lock onto the incorrect edge. One method of avoiding data association errors is to use a conservative motion model (and hence a small search range) but this is not completely effective, and would anyway be counterproductive here. In [5] a RANSAC step eliminates outliers and here we use an M-Estimator with a similar goal, but it would be preferable not to include outlier measurements in the first place.

We attempt to make edgelet measurements more reliable by building a per-keyframe database of visibility information. This is similar in spirit to the precomputed view-sphere originally employed by Harris [17] but we go beyond a binary visible/invisible flag, and our table also has an extra dimension of motion blur. For each edgelet, for each keyframe, and for each level of motion blur, we store binary visibility, max search length r_{search}, and expected edgelet response t_{min}. This table is built by the map-making thread. If an edgelet cannot be observed in a keyframe it is considered occluded from that view. On the other hand if an edgelet has been measured in a keyframe, the map-maker simulates the effects that various levels of motion blur would have on this edgelet, and records edge detection results for each case.

For each measured edgelet, the one-dimensional signal along the edgelet normal is extracted. To simulate motion blur, this is signal is convolved with box kernels of odd lengths from 3 to 65 pixels. For each blurred signal, the response to a cross-convolution with the appropriate sawtooth kernel is calculated. Each response is initially checked for presence of a maximum at the correct (edgelet-centered) location. Edges which are close to other edges will exhibit an off-center maximum with increasing blur length, and therefore the maximum level of blur under which such edgelets may be measured is recorded. Next, the edge response level t_{max} at the maximum is recorded, and we set $t_{min} = 0.5\,t_{max}$. Finally, the distance r_{near} to the nearest local maximum with response greater than t_{min} is recorded, and we set $r_{search} = 0.5\,r_{near}$.

6 Image-Based Motion Initialisation

This section describes a rapid method of estimating camera rotation between two frames. The method we use makes some simplifying assumptions that 1. the camera is purely

rotating between frames, 2. either a telephoto lens is used or a spherical projection, 3. the exposure setting does not change much between frames and 4. there is no moving clutter. (None of these assumptions is valid in our application, but in practice only moving clutter becomes a problem.)

To estimate inter-frame rotation, we directly compare subsampled and blurred images of the two frames. Each incoming frame is already subsampled down to 80×60 pixels for tracking large-scale point features. We subsample this one octave further to 40×30 pixels and apply a Gaussian blur of $\sigma = 0.75$ pixels. The image thus formed for each new frame is next aligned to that of the previous frame so as to minimise the sum-squared-difference over the whole image. This is done by direct second-order minimisation [19]. We minimise over the three-parameter group SE(2) in image (pixel) space, allowing ten iterations for convergence. Finally, the resulting 3-DOF image-space transformation is converted to a best-fit 3D camera rotation by considering the motion of a few virtual sample points placed about the image, in a procedure similar to the unscented transform.

The blur parameter σ is a tuning variable: We find that increasing blur extends the convergence range, but also makes the method more susceptible to moving clutter.

7 Keyframe-Based Relocalisation

A failure-recovery system of some sort can be considered an essential requirement for any tracking system to be used outside of lab conditions. The underlying SLAM implementation described in [6] incorporates the recovery system of Williams et al [4], which performs relocalisation by training a fast classifier with map feature points. These map points can then quickly be detected when the system is lost, allowing pose recovery through the three-point-pose algorithm. Williams et al reported on an EKF-based SLAM system whose maps rarely exceeded 150 features, with a classifier storage requirement of 1.3 megabytes per map point. Here we operate with a map size of up to 15,000; scaling the classifier to this map size is clearly not practical. Even using only a fraction of the map points for relocalisation, the processing and memory requirements of the randomised list classifier become a concern.

Instead we exploit the fact that the SLAM system stores full keyframes, and relocalise directly from these. Instead of extracting some form of interest points and descriptors from keyframes and then matching a novel view against them, we find that keyframes are sufficiently densely distributed that the full image can be used as a descriptor: for each keyframe added to the map, we generate a sub-sampled 40×30 pixel image, apply a Gaussian blur of $\sigma = 2.5$ pixels, and finally subtract the mean image intensity. This zero-mean, heavily blurred image forms the keyframe's descriptor.

When tracking is lost, each incoming video frame is similarly subsampled, blurred, and mean-normalised. Next, the sum-squared-difference of this incoming image is compared to all keyframes. The camera pose is then set to the position of the keyframe with the lowest image difference. Finally, a camera rotation is estimated using the same procedure as in Sect. 6, but with extra degree of freedom to account for illumination variation: Minimisation is over the three SE(2) parameters and a mean intensity offset. Camera pose is rotated by the result of this alignment and normal tracking is re-started.

8 Implementation Notes

The motion initialisation algorithm is run at the beginning of every frame, and completely replaces the SLAM system's previous motion model. After this, points and edgelets are tracked jointly in two stages. In an initial coarse stage, only the highest-level point features and edgelets with a large (20+ pixels) expected search range are measured. After a pose update from these measurements further point features and edgelets are also measured in a fine stage. We currently employ no accurate estimate of measurement uncertainty: For points and edgelets in both tracking and bundle adjustment, measurements are assumed to have an independent single-pixel variance.

An important part of a relocalisation system is a mechanism to detect tracking failure. Such a mechanism must strike a balance between promptly detecting failure when it occurs and being lenient enough to permit tracking through difficult motions. Here we employ the same heuristic check as [6] – this is based on the ratio of point features expected to point features actually measured – however we suspend failure detection during rapid camera motion.

9 Results

9.1 Edgelets in the SLAM Map

Figure 4 shows a map of a desk generated by the system described above. The distribution of points to edgelets is fairly typical: Points outnumber edgelets by a factor of 5 to 1. This is partially explained by the fact that point features can exist at many scales,

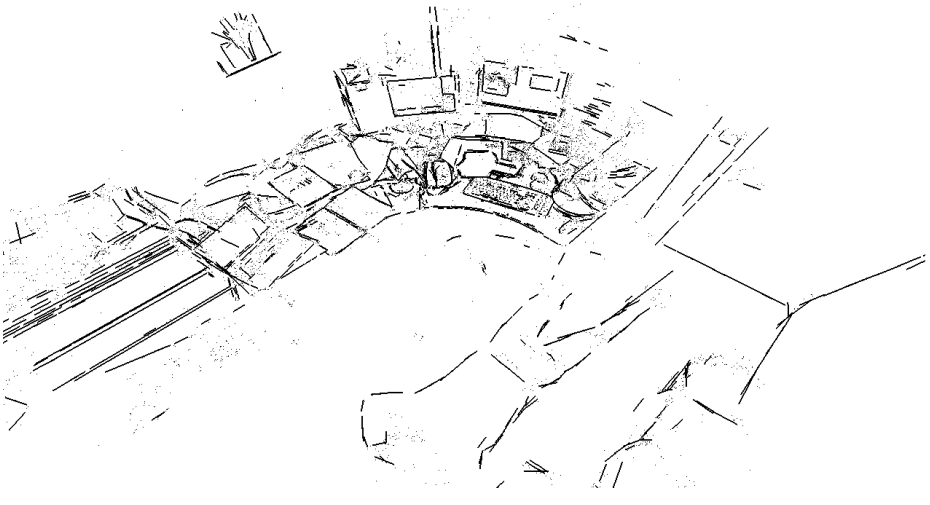

Fig. 4. A 3D map of an office scene with 9400 point features, 1800 edgelets and 250 keyframes. Some structure is clearly visible, including a door-frame, cabinet, resting researcher and a desk with some computer screens.

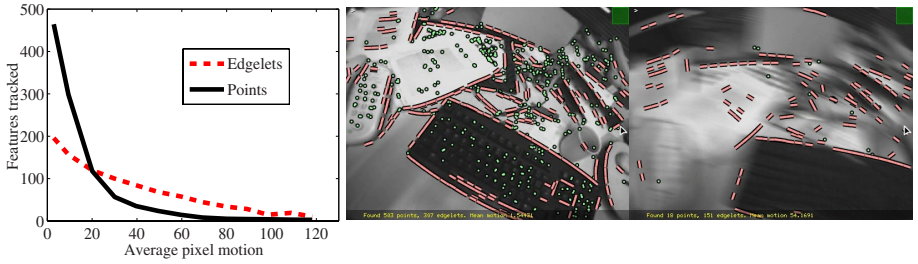

Fig. 5. The effect of image motion of number of features trackable. Left, average plots from 3500 frames. Center, in a stationary frame, 503 points and 307 edgelets are tracked. Right, with ∼54 pixels of motion, 18 points and 151 edgelets are tracked.

and also that our method for adding edgelets to the map is comparatively conservative (for example, points are not checked against a third keyframe).

A slight change in user behaviour is required to properly populate the map with edges: due to the aperture problem, a baseline in two orientations must be provided to map all edges. In practice this means that exploration is now best performed with a sinusoidal or zig-zag camera motion, rather than a linear one.

9.2 Timings

On a 2.6 GHz dual-core computer, frame-to-frame processing for the illustrated map requires 12ms per frame when tracking only point features and 18ms when tracking points and edgelets. The frame-to-frame rotation initialisation is fast and adds only 0.5ms of processing time per frame. When adding a new keyframe to the map, performing edge extraction and adding edgelets can require up to 20ms per keyframe. This is twice as long as it takes to add points, but anyway pales compared to the cost of bundle adjustment, to which edgelets add a 25% extra processing burden. Edgelets are therefore individually more expensive than point features.

9.3 Agility

It is difficult to convey the behaviour of a real-time tracking system on paper, so we encourage the reader to refer to the attached results video which demonstrates the operation of the system. Subjectively, we note three changes in system behaviour: The first is an increased robustness to short and sharp camera pans, similar to a saccade of the eyes. Where the unmodified system would fail to track this and relocalise, the motion initialisation stage is able to track these sharp rotations with sufficient accuracy for the point tracker to converge at the end of the saccade. The addition of edge features is not necessary to support these motions. The second change is an improved ability to track fast linear translations perpendicular to the camera axis. For these motions, continual tracking is necessary and the addition of edge features beneficial. The final change in behaviour is negative: The motion initialisation method is susceptible to moving clutter

in the image, such as a hand moved rapidly before the camera. Such moving clutter can cause tracking to fail, or (worse) shift tracking to a local minimum if observing repeated texture. These behaviours are illustrated in the accompanying video file.

Figure 5 illustrates the effect of rapid camera motion on tracking ability, sampled from 3500 frames. When the camera is moved slowly, point feature measurements outnumber edgelet measurements (as expected, since point features outnumber edgelets). As the camera is moved with increasing speed, causing motion blur, the number of point features measured drops rapidly below the number of available edgelet measurements.

9.4 Keyframe-Based Relocalisation

For a map with 250 keyframes, the new relocalisation method requires 1ms to find the best-match keyframe, and a further 0.5ms to calculate a rotation in this keyframe. This compares favourably with our previously used relocalisation method [4], which when used with 256 feature classes required up to 20ms for classification alone.

The methods differ in that our keyframe-based approach is not invariant to large rotation: While relocalisation is possible with a camera tilted up to circa 35°, full upside-down relocalisation from a novel view is not (but the utility of such a capability may be questioned). A further difference is that the keyframe-based method is more predictable in operation: Both methods occasionally fail to relocalise, and when this happens the new method guarantees relocalisation if the user simply moves the camera back to a 'known-good' location. By contrast the method of [4] may require a more random search pattern.

10 Conclusions and Further Work

This paper has demonstrated how the tracking agility of a monocular SLAM system can be improved. By combining and enhancing techniques from monocular SLAM, model-based 3D tracking and direct image-based tracking, we achieve an unprecedented level of robustness towards rapid camera motions. It is however important to distinguish between tracking agility and mapping agility: While we can track the camera during rapid motions exhibiting blur, such frames are not used for map-building. If anything, the rate at which a user can expand the map has slowed. The need for orthogonal baselines and three keyframes makes addition of edgelets to the map slower than that of points – incremental systems such as [10] can (given benign conditions) add new features more quickly. A combination of techniques may be an interesting area of future work.

Finally, the system makes no attempt to use edgelets beyond tracking (and bundle adjustment): No attempt is made to link them into longer constructs or use them to obtain occlusion information for points. While the addition of edgelets to the map makes this more visually useful to a human operator, future work should investigate how a computer can better exploit the rich geometric information available.

Acknowledgements

This work was supported by EPSRC grant GR/S97774/01.

References

1. Davison, A., Reid, I., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Trans. Pattern Analysis and Machine Intelligence 29, 1052–1067 (2007)
2. Chekhlov, D., Pupilli, M., Mayol-Cuevas, W., Calway, A.: Real-time and robust monocular SLAM using predictive multi-resolution descriptors. In: Proc 2nd International Symposium on Visual Computing (November 2006)
3. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–100 (2004)
4. Williams, B., Klein, G., Reid, I.: Real-time SLAM relocalisation. In: Proc 11th IEEE International Conference on Computer Vision (ICCV 2007), Rio de Janeiro (October 2007)
5. Eade, E., Drummond, T.: Scalable monocular SLAM. In: Proc. IEEE Intl. Conference on Computer Vision and Pattern Recognition (CVPR 2006), New York, pp. 469–476 (2006)
6. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: Proc Intl. Symposium on Mixed and Augmented Reality (ISMAR 2007), Nara (November 2007)
7. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951. Springer, Heidelberg (2006)
8. Klein, G., Drummond, T.: Tightly integrated sensor fusion for robust visual tracking. In: Proc. British Machine Vision Conference (BMVC 2002), Cardiff, pp. 787–796 (September 2002)
9. Smith, P., Reid, I., Davison, A.: Real-time monocular SLAM with straight lines. In: Proc British Machine Vision Conference (BMVC 2006), Edinburgh (September 2006)
10. Eade, E., Drummond, T.: Edge landmarks in monocular SLAM. In: Proc. British Machine Vision Conference (BMVC 2006), Edinburg (September 2006)
11. Viéville, T., Faugeras, O.D.: Feedforward recovery of motion and structure from a sequence of 2d-lines matches. In: Proc. 3rd Int. Conf. on Computer Vision, pp. 517–520 (1990)
12. Faugeras, O.D., Lustaman, F., Toscani, G.: Motion and structure from point and line matches. In: Proc. 1st Int. Conf. on Computer Vision, London, pp. 25–33 (1987)
13. Taylor, C.J., Kriegman, D.J.: Structure and motion from line segments in multiple images. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(11), 1021–1032 (1995)
14. Bartoli, A., Sturm, P.: Structure from motion using lines: Representation, triangulation and bundle adjustment. Computer Vision and Image Understanding 100(3), 416–441 (2005)
15. Shashua, A.: Trilinearity in visual recognition by alignment. In: Proc. 3rd European Conf. on Computer Vision, Stockholm, May 1994, pp. 479–484. Springer, Heidelberg (1994)
16. Devernay, F., Faugeras, O.D.: Straight lines have to be straight. Machine Vision and Applications 13(1), 14–24 (2001)
17. Harris, C.: Tracking with rigid models. In: Blake, A. (ed.) Active Vision, MIT Press, Cambridge (1992)
18. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 8(6), 679–698 (1986)
19. Benhimane, S., Malis, E.: Homography-based 2d visual tracking and servoing. Special Joint Issue on Robotics and Vision. Journal of Robotics Research 26(7), 661–676 (2007)