# Fusing Points and Lines for High Performance Tracking

Edward Rosten and Tom Drummond
{er258|twd20}@eng.cam.ac.uk
Department of Engineering, University of Cambridge
Cambridge, CB1 2BZ, UK

## Abstract

*This paper addresses the problem of real-time 3D model-based tracking by combining point-based and edge-based tracking systems. We present a careful analysis of the properties of these two sensor systems and show that this leads to some non-trivial design choices that collectively yield extremely high performance. In particular, we present a method for integrating the two systems and robustly combining the pose estimates they produce. Further we show how on-line learning can be used to improve the performance of feature tracking. Finally, to aid real-time performance, we introduce the FAST feature detector which can perform full-frame feature detection at 400Hz. The combination of these techniques results in a system which is capable of tracking average prediction errors of 200 pixels. This level of robustness allows us to track very rapid motions, such as 50° camera shake at 6Hz.*

## 1. Introduction

While substantial advances have been made in the domain of real-time visual tracking, there is still a need for systems which can tolerate the very rapid translations, rotations and accelerations that occur in unconstrained handheld use in scenes with substantial unmodeled clutter. These motions lead to large prediction errors and to cope with this, we propose a method for combining the two popular approaches to tracking, namely point-based and line-based tracking.

These two techniques have very different statistical properties; they have different failure modes, exhibit different error structures and have differing requirements for their operation. By analysing these properties, we show that careful integration can achieve more than simply combining the two systems in a big linearizing filter (such as an EKF). This analysis is presented in Section 2 in terms of the *preconditions* and *postconditions* of each of the two approaches. Their combination is described in Section 3 and the statistical techniques required are detailed in Section 4.1. We also present a practical contribution, in the form of an extremely efficient feature detector and matching system in Section 5. Results from the combination of these techniques, are presented in Section 6.

### 1.1. Background

Point features have strong characteristics and this makes it relatively easy to localize them and to find correspondences between frames. This makes point-based systems robust to large, unpredictable inter-frame motions. However, over several frames the appearance of a feature point can change substantially, due to the change in scene structure relative to the camera. Although features which are invariant to scaled Euclidean [11] or affine [13] transformations exist, it is not always advantageous to use them because they can be expensive to compute and the invariances reduce their ability to discriminate [17]. These changes in appearance can lead to matching errors and drift in the pose estimate over time.

Point features can be used with [18] or without [2] a surface model. Using a surface model means that larger numbers of feature points can be used to assist pose estimation while retaining real-time performance: given a pose estimate, their 3D position can be obtained by back projection onto the surface. However, errors in the position of the surface create 3D point position errors which lead to an amplification of pose drift.

One method for removing (or reducing) drift is presented in [18] which couples the surface model with the use of keyframes to provide a global database of the 3D position and appearance of feature points. However, populating a large immersive 3D environment with a sufficient number of keyframes would be both difficult and very time consuming. Further, in many environments, the clutter which gives rise to most feature points (e.g. the clutter on desks in an office environment) changes over medium — to long — term time scales, even if the surface model remains approximately correct. The system presented here instead uses line features to avoid drift.

Line features [5, 1, 10] and their descriptor (a gradient

in intensity) are stable under a very wide range of lighting conditions and aspect changes. It is precisely this invariance which makes the feature non discriminative, and it is therefore difficult to find correspondences. As a result it is difficult to track edges robustly.

The usual solution — assuming that the closest edge to the expected edge position is the correct match — can lead to a large number of correspondence errors if the motion is large, although [4] alleviates this somewhat by using a robust estimator. [12] operates by first finding the 2D motion of the object, and then the full 3D pose, once the object is in nearly the correct place. The approach is also adopted by [9] and [17], who both find the 2D motion of the object by matching features between frames and then use this to initialize an edge based tracker. [9] uses a simple combination of measurements from the feature tracker, which minimizes a weighted sum of edge errors and feature errors.

## 2. Sensor analysis

In order to combine these two approaches for pose estimation, it is first necessary to look in more detail at the *preconditions* and *postconditions* for each in order to understand the difference in statistical behavior. We present first a summary of these in order to provide a reference for the discussion that follows.

| | |
|---|---|
| Point based tracking | |
| Preconditions | |
| 3D point cloud/model. | $\mathcal{P}_1^-$ |
| Postconditions | |
| Produces robust differential measurements... | $\mathcal{P}_1^+$ |
| ...with approximately Gaussian posterior. | $\mathcal{P}_2^+$ |
| Posterior measurement covariance is inaccurate. | $\mathcal{P}_3^+$ |
| Edge based tracking | |
| Preconditions | |
| Geometric 3D edge model. | $\mathcal{E}_1^-$ |
| Accurate pose prior. | $\mathcal{E}_2^-$ |
| Postconditions | |
| Non-Gaussian posterior. | $\mathcal{E}_1^+$ |
| Drift-free measurement. | $\mathcal{E}_2^+$ |

### 2.1. Point features

**Condition** $\mathcal{P}_1^-$: In Section 1.1 we argued that obtaining a static point cloud for large scenes is unfeasible. As a result, it it necessary to dynamically learn this model and this is achieved by back-projection onto a geometric 3D surface model. Because there are no static features in this point cloud, the tracker can only produce differential pose measurements (**Condition** $\mathcal{P}_1^+$).

**Condition** $\mathcal{P}_2^+$ arises because we can determine which point matches are correct and which are not with high probability (see Section 4.1 and Section 4.2). The measurement errors of the inliers are mostly due to pixel quantization (we do not use a sub-pixel feature detector—see Section 5) and so are independent. The likelihood therefore approaches a Gaussian distribution by the central limit theorem.

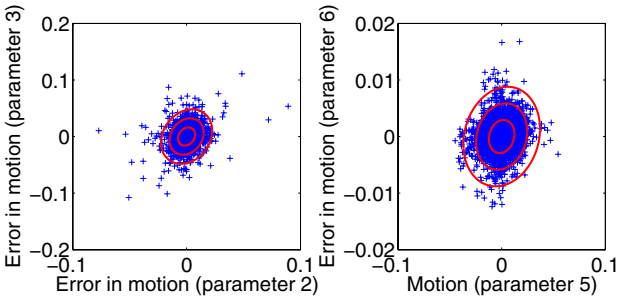


Figure 1: The errors between the point tracking posterior and the ground truth are well modeled by uncorrelated statistics. To demonstrate this, the two strongest correlations have been shown and even these are only weakly correlated.

**Condition** $\mathcal{P}_3^+$: although the measurement errors are independent, the errors in the 3D point cloud are not. Points detected on unmodeled structural clutter are back-projected on to the closest modeled plane, which is almost always further from the camera than the clutter. The result is that any errors in the 3D point cloud may well be strongly correlated, and it is therefore we find that the covariance obtained from the point matches is inaccurate. As a result we must model the covariance. We consider two models for the covariance. The first model is that the covariance, $\mathbf{C}$, can be modeled as a function of the motion, $\boldsymbol{\mu}$:

$$\mathbf{C} = \mathbf{A} + \mathbf{B}\boldsymbol{\mu}\boldsymbol{\mu}^{\mathsf{T}}\mathbf{B}^{\mathsf{T}} \tag{1}$$

We test this by using data which is obtained from a sequence where the pose is found by manual alignment in each frame. We find that $\mathbf{A}$ is largely diagonal, and $\mathbf{B}$ consists of only very small values. The data corresponding to the largest off-diagonal element of $\mathbf{A}$ and the largest element of $\mathbf{B}$ is shown in Figure 1. The second model we consider assumes that the shape of the covariance obtained from the data (see Section 4.1) is correct, but that it is over-saturated by a constant, $k$. The most likely value of $k$ maximizes the log-likelihood of the data and is given by:

$$k = \underset{k}{\operatorname{argmin}} \left( -\sum_i \mathbf{e}_i{}^{\mathsf{T}} (k\mathbf{C}_i)^{-1} \mathbf{e}_i - \ln \sqrt{(2\pi)^6 |k\mathbf{C}_i|}, \right) \tag{2}$$

where $\mathbf{C}_i$ is the computed covariance for frame $i$ and $\mathbf{e}_i$ is the 6 DOF pose error for frame $i$, obtained from the ground truth data. We find that $k \approx 7200$.

### 2.2. Edge tracking

**Condition** $\mathcal{E}_1^-$: in order to perform edge tracking we must have a 3D edge model of the object to be tracked

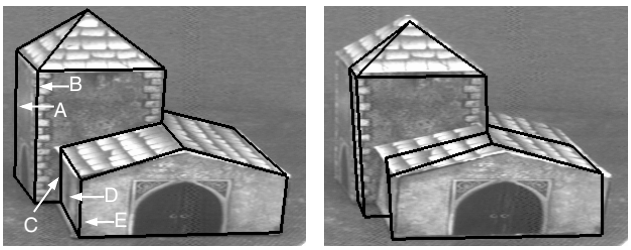


Figure 2: Left: model in the correct position. Right: edge tracking fails because model edges A and B lock on to image edge B and model edges C and E lock on to image edge D.

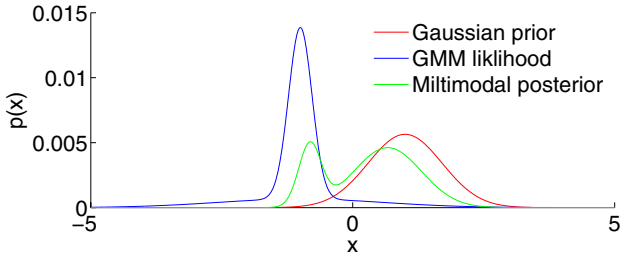


Figure 3: A 1 dimensional example of a Gaussian prior combining with a mixture model likelihood to produce a multimodal posterior.

(this is the model on to which features are projected for **Condition** $\mathcal{P}_1^-$). The model is created by hand. Edge features are invariant to lighting and perspective changes, hence the model can remain static. Because we have this absolute model the measurements obtained will be drift-free(**Condition** $\mathcal{E}_2^+$). Highly invariant features are not discriminative, so in order to avoid incorrect edge correspondences, we require a strong prior on model pose, and hence image edge position (**Condition** $\mathcal{E}_2^-$). Even with this, edges can still be detected incorrectly, as illustrated in Figure 2. These incorrect measurements will often be strongly correlated and hence the pose estimate will contain a large error. The correlation in the error means that the posterior pose does *not* approach a Gaussian by the central limit theorem (**Condition** $\mathcal{E}_1^+$). We therefore model this distribution as a two component GMM (Gaussian Mixture Model), consisting of a narrow Gaussian (the distribution of poses where the correspondences are correct) and a very broad Gaussian (the distribution when edge correspondences are incorrect).

## 3. Sensor fusion

Both feature based and edge based tracking have failure modes, but these are complementary and so combining them leads to a more robust system. Because of the non-Gaussian statistics of the system, measurements can not be trivially fused by using linear techniques such as Kalman filtering, so several strategies are needed to robustly combine the measurements. The above analysis leads to the following conclusions:

1. Points are robust to large motions (**Condition** $\mathcal{P}_1^+$), and lines need reasonably accurate initialization (**Condition** $\mathcal{E}_2^-$) hence points should be treated first and lines second.

2. The statistics of line measurement are non-Gaussian (**Condition** $\mathcal{E}_1^+$) so a non-linear filter is needed

3. Under large aspect changes, point appearance can change substantially, so a system which estimates inlier probability for each point would be beneficial.

The pose estimate covariance $\mathbf{C}_a$ and point cloud (**Condition** $\mathcal{P}_1^-$) from the previous frame are used. The point tracker adds a differential measurement (**Condition** $\mathcal{P}_1^+$) resulting in a posterior covariance $\mathbf{C}_b = \mathbf{C}_a + k\mathbf{C}_f$, where $\mathbf{C}_f$ is the covariance measured by the feature point tracker (see Section 4.1).

The brittle, but precise edge tracker is initialized (**Condition** $\mathcal{E}_2^-$) using robust differential measurements from **Condition** $\mathcal{P}_1^+$, as in [8]. We use the tracker described in [4], but with edge threshold used is normalized by image intensity. We also perform multiple iterations of the tracker per frame, which is similar to the system described in [9], except that we linearize between iterations. Since the likelihood of the edge based tracker is a two-component GMM (**Condition** $\mathcal{E}_1^+$) and the prior is a Gaussian, the posterior, $p_{\text{post},c}$ is also a GMM which may have two modes (see Figure 3). This posterior for pose is then used to obtain the 3D point cloud needed for the next frame. Since the posterior can be bimodal, a separate point cloud is generated for each mode. Note that if the edge tracker is correct, this estimate of posterior pose is drift-free (**Condition** $\mathcal{E}_2^+$).

If both modes were to be propagated all the way through the next iteration, exponential mixture component explosion would follow since edge tracking doubles the number of mixture components at each iteration This is avoided by comparing the performance of each point cloud on the subsequent point tracking stage. The GMM that gave rise to the point clouds gives their relative probability, while the difference in residual error in point tracking provides an estimate of their likelihoods. These are then combined and only the Gaussian component (and associated point cloud) with the highest posterior probability is retained for the edge tracking stage. The algorithm described above is summarized below:

1. A new frame arrives and point features are detected.
2. Correspondences are found between the new features and existing features on the model.
3. The probability that a match is correct is computed from the correspondence score.

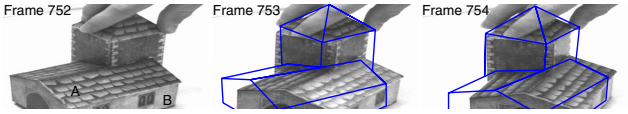


Figure 4: Three consecutive video frames are shown while the model rotated rapidly (about $720^\circ/s$) around its centre. The outline indicates the position of the model in the previous frame. Features on face A change appearance and features on face B change shape significantly.

4. The pose is robustly computed for both modes, and the most probable mode is kept.
5. The new pose is used to initialize and run an edge tracker.
6. The features are back-projected on to the model.
7. The learned relationship between matching score and matching probability is updated based on the posterior match probability.

We use a calibrated camera (with modeled radial distortion). The test system used for timings is a P4 Xeon at 2.4GHz. The video sequences consist of $768 \times 288$ pixel fields from a 50Hz PAL source.

# 4. Feature tracking

## 4.1. Position optimization

Under large frame-to-frame motions, such as the one shown in Figure 4, feature points can change appearance significantly and this typically leads to a large number of mismatched features. In some sequences, consecutive frames have as few as 10% of points matched correctly. Although we are using the crude (but fast) point detector and matcher detailed in Section 6, points from these frames were analysed using SIFT[11] and a similar matching percentage was obtained.

In classical Bayesian fashion, the most likely model parameters are computed by maximising the probability of the observed data given the parameters (multiplied by a weak prior over the parameters). In frame $n$, we have two sets of features: one set of features on the model, $F_{O,n}$ (extracted in the previous frame, and reprojected under a motion $\boldsymbol{\mu}$), and another set, $F_{I,n}$, which have been extracted from the image. Between these features, we have a set of matches $M = \{m_1, ..., m_N\}$, where a match is given by $m_i = \{f_{O,i}, f_{I,i}\}$, where $f_{O,i} \in F_{O,n}$ and $f_{I,i} \in F_{I,n}$.

The set of matches can be regarded as being made up of correct matches, $M_\Upsilon$, and incorrect matches, $M_\Phi$.

If $m_i \in M_\Upsilon$, then $f_{I,i}$ is in the same place that $f_{O,i}$ projects to under the motion to be recovered, with some added measurement noise. If $m_i \in M_\Phi$ then $f_{I,i}$ can appear anywhere in the image, with approximately uniform probability. If $\mathbf{e}_i$ is the Euclidean distance between $f_{O,i}$ and $f_{I,i}$ in the image, then the PDF of observing the matches is given by:

$$p(M|\boldsymbol{\mu}) = \prod_{i=1}^{N} \frac{(1-\alpha)}{A} + \alpha \frac{e^{(-\frac{\mathbf{e}_i^{\mathsf{T}}\mathbf{e}_i}{2\sigma^2})}}{2\pi\sigma^2}, \qquad (3)$$

where $A$ is the image area, $\alpha$ is the expected proportion of good matches and $\sigma$ is the measurement noise. In theory, one method for finding the most likely $\boldsymbol{\mu}$ is to use iterative reweighted least squares (IRWLS) where the reweighting function is the posterior probability that a match is an inlier.

In practice, with a large number of mismatches, IRWLS will often not succeed because of local maxima in the likelihood. We will demonstrate this with a one-dimensional example. We start by placing points randomly on the unit circle, giving $F_O$. The model is then rotated by $\boldsymbol{\mu}$ radians, which is the parameter to be determined. To simulate matching, the new positions of the points are either corrupted by Gaussian noise with variance $\sigma^2$, or scattered about the unit square (Figure 5), giving $F_I$.

The likelihood of the data given $\boldsymbol{\mu}$ and the correct value of $\sigma_g$ is shown in Figure 5 and visibly contains *many* local maxima with the absolute maximum being a very narrow peak. Hence, in order to find the maximum, it is necessary to use a technique which is robust to local maxima. Generalized Monte-Carlo based techniques such as simulated annealing[7] escape local optima easier by randomly

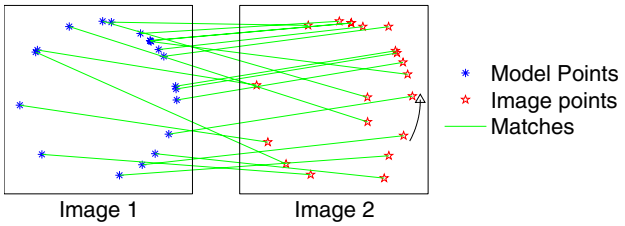


Figure 5: A synthetic example of feature matching between frames, where the model has a single parameter. Only a small fraction of the data is shown.

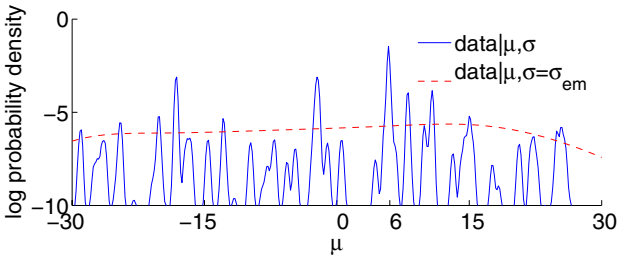


Figure 6: Probability density of observing $M$ for the one dimensional example. PDFs are shown for the known value of $\sigma$, $\sigma_g$ and where EM converges, $\sigma_{em}$, at the known value of $\alpha$. The correct value of $\boldsymbol{\mu}$ is 6.

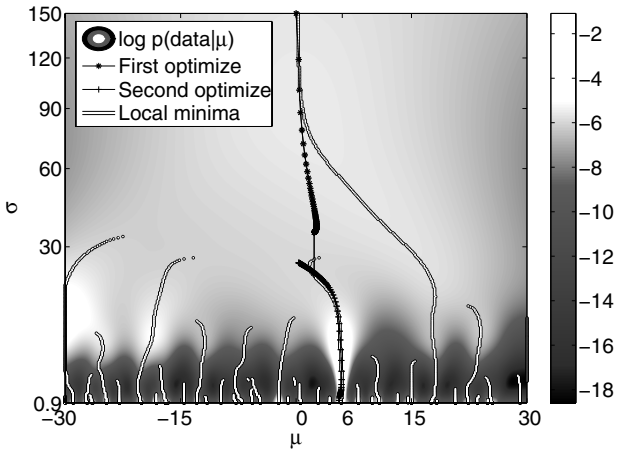


Figure 7: The greylevel plot shows the likelihood of the data given the model parameter, $\boldsymbol{\mu}$, for different levels of blur with the known value of $\alpha$. the bottom row in this plot is the same as the graph in Figure 6. The graph also shows the path that EM takes through this space. The calculated values for $\alpha$ are not shown.

perturbing the system. Perturbations are accepted with a probability based on how energetically favorable it is to accept the perturbation. At higher temperatures, the cost function is raised to a small power, making perturbations more likely to be accepted. As the system is annealed, the power is increased, making it harder to escape. However, in this case, optimum is very narrow, so there is only a very small probability that a perturbation, or indeed any particle based method, will land in the optimum. Instead, it would be preferable to make the correct peak broader by convolving the likelihood with a Gaussian to blur it. This gives the following PDF:

$$p(m_i|\boldsymbol{\mu}) = \frac{(1-\alpha)}{A} + \alpha \frac{e^{(-\frac{\mathbf{e}_i{}^{\mathsf{T}}\mathbf{e}_i}{2(\sigma^2+\sigma_b^2)})}}{2\pi(\sigma^2+\sigma_b^2)}, \qquad (4)$$

where $\sigma_b$ is the size of the blur. This is equivalent to using a value of $\sigma$ larger than the true value. Because for large $\sigma$, the peak will not be in the correct place, a schedule is needed to reduce it to its correct value and the most effective way to do this is by using the EM (Expectation Maximization) algorithm[3] to find a PDF which fits the data. This is convenient because it gives us a framework for calculating unknown variables such as the true value of $\alpha$ and $\sigma$. This is illustrated in Figure 7.

In the presence of large amounts of noise, the EM algorithm can converge on a solution where $\sigma$ is too large, as shown in Figure 6. Although the precise value of $\sigma$ is unknown, we know its approximate value; a large proportion of the measurement error comes from pixel quantization, hence $\sigma \sim 1\text{pixel}$. If EM converges to $\sigma^2 \gg 1$, then the optimization is given a 'kick' downwards by forcibly lowering $\sigma$, and recomputing $\alpha$. The effect of this is shown in Figure 7.

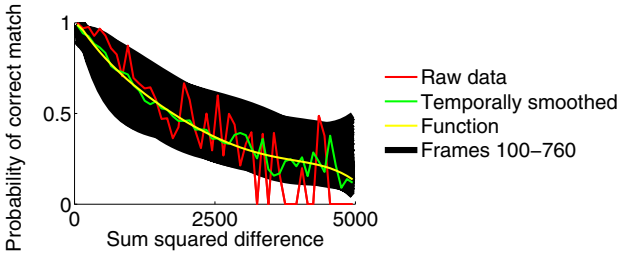


Figure 8: Graphs showing the function which maps SSD to match probability. The function, temporally smoothed and raw data are given for frame 200.

EM jointly optimizes the posterior probability $P(m_i \in M_\Upsilon)$ for each point. Since we have a prior estimate of this from the matching score (see Section 4.2) the posterior becomes:

$$P(m_i \in M_\Upsilon) = \frac{P_p \alpha p_\Upsilon}{\frac{(1-P_p)(1-\alpha)}{A} + P_p \alpha} \qquad (5)$$

where

$$P_p = P_{\text{prior}}(m_i \in M_\Upsilon)$$

and

$$p_\Upsilon = \frac{e^{(-\frac{\mathbf{e}_i{}^{\mathsf{T}}\mathbf{e}_i}{2(\sigma^2+\sigma_b^2)})}}{2\pi(\sigma^2+\sigma_b^2)}.$$

The covariance of the posterior pose is:

$$\mathbf{C}_f = \left( \sigma^2 \sum_i P_{\text{post}}(m_i \in M_\Upsilon) \mathbf{J}_i{}^{\mathsf{T}} \mathbf{J}_i \right)^{-1}, \qquad (6)$$

where $\mathbf{J}$ is the Jacobian relating image motion (in pixels) of a point to the motion parameters.

We note that Guided[15] MLESAC[16] could be used here instead of EM, and would take in the same prior information and produce an equivalent posterior. It is unclear which would be computationally more efficient, but the current system is sufficient for real-time operation.

### 4.2. Calculation of the match prior from SSD

For a given feature point in $F_O$, $f_n$, its correspondence is calculated by finding the feature point in $F_I$ which minimizes the sum squared difference (SSD) of the feature vector between the feature points. The SSD is a measure of difference in appearance of the two feature points, and we make the intuitive assumption that of all the features in

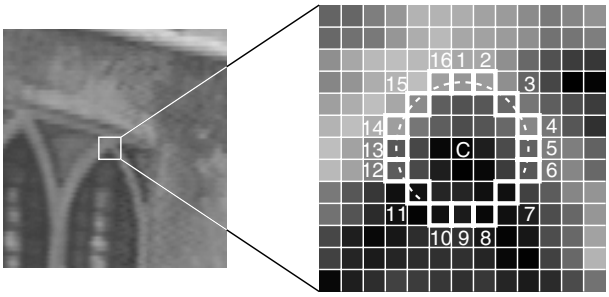


Figure 9: FAST Feature detection in an image patch. The highlighted squares are the pixels used in the feature detection. The pixel at C is the centre of a detected corner: the dashed line passes through 12 contiguous pixels which are brighter than C by more than the threshold.

frame $n+1$, those that look most like $f_n$ (have the smallest SSD) are the most likely to be correctly matched. In other words, there is likely to be a relationship between the SSD and the probability that the feature match represents an inlier. We make explicit use of this relationship by on-line learning of a dynamic function which maps SSD to inlier probability.

One outcome of the EM algorithm used in Section 4.1 is the posterior probability that each match is correct. This information along with the SSD values for each match can be used to provide an estimate of the relationship between SSD and inlier probability for the next frame. This is achieved by binning the SSD scores and computing the average inlier probability for each bin. To compensate for the limited amount of data in each frame (especially for large SSD values), this data is temporally smoothed with a time constant of 10 frames. A cubic polynomial is then fitted to the resulting values and this is used as the SSD to inlier probability function for the next frame. This technique provides a dramatic improvement in the performance of the point tracker, and the results of this can be seen in Figure 8. This figure also illustrates the substantial range of possible functions that can be generated over time, hence the necessity to use dynamic learning at each frame.

## 5. FAST feature detection and matching

The robust optimization described in Section 4.1 allows us to trade off the quality of corner detection and matching for speed. We present here the FAST (Features from Accelerated Segment Test) feature detector[1]. This is sufficiently fast that it allows on-line operation of the tracking system. A test is performed for a feature at a pixel $p$ by examining a circle of 16 pixels (a Bresenham circle of radius 3) surrounding $p$. A feature is detected at $p$ if the intensities of at least 12 contiguous pixels are all above or all below the intensity of $p$ by some threshold, $t$. This is illustrated in Figure 9. The test for this condition can be optimized by examining pixels 1, 9, 5 and 13, to reject candidate pixels more quickly, since a feature can *only* exist if three of these test points are all above or below the intensity of $p$ by the threshold. With this optimization, on a sample sequence of video, the algorithm presented examines, on average, 3.8 pixels to test if there is a feature at a given location.

This type of corner detection naturally leads to using the pixel intensities from the 16 pixel circle as a feature vector. It further follows that features can be categorized as *positive* (where the pixels are greater than the center) and *negative*. This partitioning is useful since positive features need not be compared to negative ones.

The result of corner detection on a typical frame is shown in Figure 10. In Table 1 we present the speed of the FAST feature detector for a 'typical' sequence. The number of features detected and hence the speed of detection is deter-

[1]Source code available in the libcvd library, hosted on `http://savannah.nongnu.org/projects/libcvd`

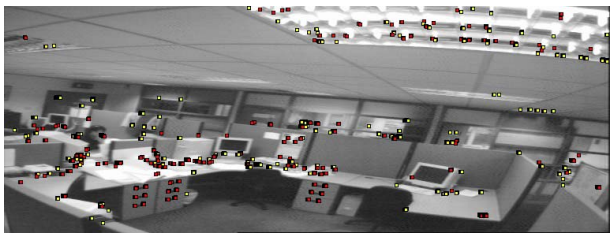

Figure 10: FAST Feature detection in a typical frame. Positive corners are indicated by ■ and negative corners are indicated by ■. The threshold is 25.

| Detector | FAST | SUSAN | Harris |
|---|---|---|---|
| Time (ms) | 2.6ms | 11.8ms | 44ms |

Table 1: Time taken to perform feature detection on a PAL field ($768 \times 288$ pixels) on the test system. The SUSAN[14] detector used is the reference implementation. The Harris[6] detector code was optimized to use the SSE vectorising instructions.

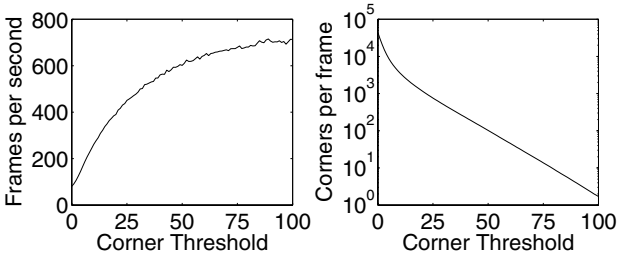


Figure 11: Graphs showing how the speed of corner detection and number of corners vary with the corner threshold. Because the corner detector is able to reject unsuitable candidates quickly, the speed is related to the number detected.

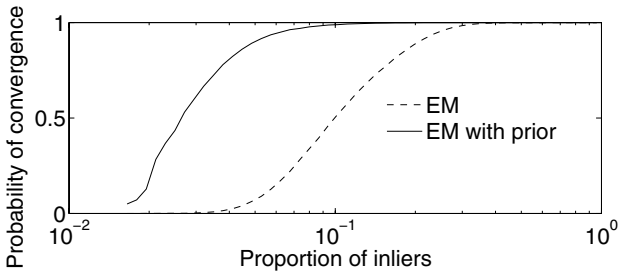


Figure 12: The synthetic results demonstrate that having a good estimate of $P(m_i \in M_\Upsilon)$ greatly improves the probability of a successful convergence.

mined by the threshold. The Figure 11 shows how these vary with the threshold.

### 5.1. Feature matching

For fast matching, it is important to avoid the $\mathrm{O}(N^2)$ cost of testing every feature against every other feature. To do this, $F_I$ is sorted by the mean value of the feature vectors, $\overline{f}$. When matching a feature, $f_{O,j}$, we use binary search to find the feature in $F_I$ with the closest mean, from which a linear search is performed. The the SSD is bounded by

$$SSD(f_{O,j}, f_{I,i}) \geq l\,(\overline{f_{O,j}} - \overline{f_{I,i}})^2 \qquad (7)$$

where $l$ is the number of elements in the feature vector (transforming the basis of the vectors and comparing the first elements gives this result) . This bound can be used to rapidly terminate the linear search.

The time taken to perform SSD is reduced by applying a transformation to the feature vectors which compacts most of the energy into the first few elements. This permits rapid rejection of potential matches without computing the full SSD. Each frame, a transformation which performs optimal energy compaction can be found, but any gains this might have are more than offset by the time taken to find the rotation (43ms on the test system). Further, in general, rotation can only be applied with an $\mathrm{O}(N^2)$ matrix multiplication. Instead, we use specific ones which can be applied with a fast $\mathrm{O}(N \log N)$ transform. We have tried the Discrete Cosine Transform and the Harr Wavelet Transform. The performance is very similar, but the Harr transform can be computed more efficiently.

We have found that for our typical scenes (500–1500 points, with 16 element feature vectors) this method is faster than a *k*-d tree, since very highly optimized sorting algorithms are available and computing SSD is very fast.

## 6. Results

### 6.1. Synthetic test of point tracking

The robust optimizer is tested using synthetically generated data:

1. Place the virtual model in view of the virtual camera.
2. Synthesize corner detection by scattering points about the virtual camera frame, keeping the ones which land on the model.
3. Generate a random motion.
4. Reproject the points (with quantization error) after the motion to simulate point matching.
5. Generate a probability that the match is correct. This corresponds to a probability obtained from the SSD.
6. Mismatching is simulated by making some of the points match to a random location in the image. The probability of this happening is based on the generated prior.
7. Optimize the position to find the motion.
8. Test the result against the known motion.

The random motion corresponds to a rotation of up to 15°and approximately 200 pixels of image motion due to translation. We generate 1000 virtual matches in each frame of which about 400 lie up on the model. Figure 12 shows the synthetic results. Without a good prior of $P(m_i \in M_\Upsilon)$, 10% inliers yields a 50% probability of convergence. With a good estimate, only 3% inliers are needed to yield a 50% probability of convergence. This is important because we frequently experience frames with only 10% inliers, and if the probability of convergence was only 50%, the sytem would fail frequently. Instead, it succeeds 99% of the time.

### 6.2. Tests on video sequences

The ability of the tracking system to deal with large inter-frame translations is shown in Figure 13. The two frames illustrated were tracked in isolation. The average image motion of point features between the two frames is 89 pixels, and the image motion of the edges closest to the camera is over 400 pixels.

This illustrates not only that the system is capable of dealing with large inter-frame motions, but also that it is capable of dealing with large amounts of structural clutter without failing. Figure 14 shows the tracker dealing with large inter-frame rotations, in the same scene. The average image motion throughout the scene is 79 pixels and the largest average image motion of a single frame is just over 204 pixels. In this sequence, a zero order motion model is used (no velocity information is tracked) and indeed, such a velocity model is unlikely to be of much use since the camera experiences angular accelerations of up to 88,500°s$^{-2}$.

Further results are given in the supplementary videos, `multimodal.mpg` and `lab.mpg`. `multimodal.mpg` shows tracking in the presence of many strong unmodeled edges, and also under heavy occlusions. The video shows a side -by-side comparison of a naïve method of sensor fusion (point tracking followed by line tracking) and the method described in Section 3. `lab.mpg` is an extended sequence

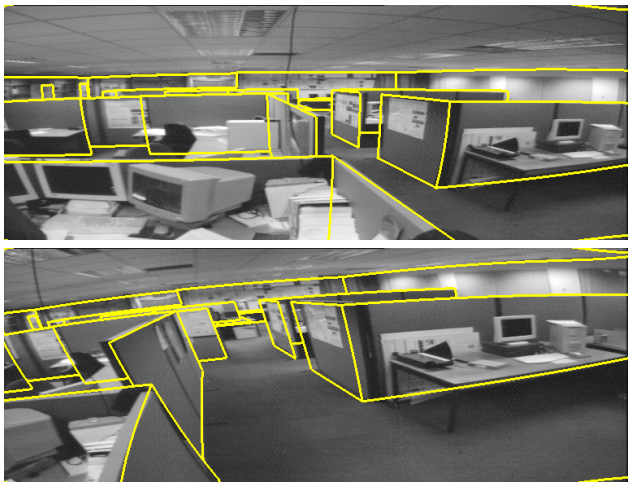

Figure 13: This shows the system tracking system coping with large translations. These consecutive frames are taken 1m apart, corresponding to a speed of 50ms$^{-1}$ (112mph). In this scene, only the vertical partitions are modeled. As a result, a large number of the visible features (such as the contents of the desks) are structural clutter. The outline of the modeled geometry is shown.

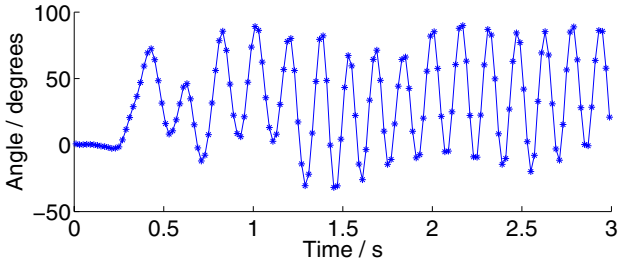


Figure 14: Graph showing the angle of a handheld camera undergoing vigorous rotational shaking at approximately 6Hz. The limiting factor is the rate and range over which the author could shake the camera.

of the scene in Figure 13 being tracked from a handheld camera. The maximum tracked speed is $12ms^{-1}$.

## 7. Summary and Conclusions

This paper presents a high performance tracking system based on the combination of two different tracking systems with complementary behavior and very different statistics. By employing a careful analysis of the requirements and behavior of these systems, their synthesis into a single system has been enhanced. This includes the use of one system to initialize the other, a non-linear method for combining the two posteriors as well as a method for on-line learning of the relationship between sum-squared difference and inlier probability for the point tracker. We have also presented a fast feature extraction and matching algorithm.

## References


[1] M. Armstrong and A. Zisserman. Robust object tracking. In *Asian Conference on Computer Vision*, volume I, pages 58–61, Singapore, 1995.

[2] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *International Conference of Computer Vision*, Nice, Oct. 2003.

[3] A. Dempster, N. Laird, and R. D.B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, B 39:1–38, 1977.

[4] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):932–946, 2002.

[5] C. Harris and C. Stennett. RAPID, a video rate object tracker. In *1st British Machine Vision Converence*, pages 73–77, Oxford, Sept. 1990.

[6] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988.

[7] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220, 4598(4598):671–680, May 13th 1983.

[8] G. Klein and T. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, Sept. 2004.

[9] P. F. L. Vacchetti, V. Lepetit. Combining edge and texture information for real-time accurate 3d camera tracking. In *International Simposium on Mixed and Augmented Reality*, Jacksonville, USA, 2004.

[10] D. Lowe. Robust model-based motion tracking through the integration of search and estimation. *International Journal of Computer Vision*, 8(2):113–122, Aug. 1992.

[11] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[12] E. Marchand, P. Bouthemy, F. Chaumette, and V. Moreau. Robust real-time visual tracking using a 2D-3D model-based approach. In *International Conference of Computer Vision*, volume 1, pages 262–268, Sept. 1999.

[13] F. Schaffalitzky and A. Zisserman. Viewpoint invariant texture matching and wide baseline stereo. In *8th International Conference of Computer Vision*, pages 636–643, Vancouver, Canada, July 2001.

[14] S. Smith and J. Brady. SUSAN - a new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, May 1997.

[15] B. Tordoff and D. Murray. Guided sampling and consensus for motion estimation. In *7th European Conference on Computer Vision*, Copenhagen, June 2002.

[16] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:138–156, 2000.

[17] B. L. V. Gouet. Sap: A robust approach to track objects in video streams with snakes and points. In *15th British Machine Vision Converence*, volume 2, pages 737–746, Kingston Upon Thames, Sept. 2004.

[18] L. Vacchetti, V. Lepetit, and P. Fua. Stable real-time 3d tracking using online and offline information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1385–1391, 2004.