# Robust feature matching in $2.3\mu$s

Simon Taylor Edward Rosten Tom Drummond
Department of Engineering, University of Cambridge
Trumpington Street, Cambridge, CB2 1PZ, UK
{sjt59, er258, twd20}@cam.ac.uk

## Abstract

*In this paper we present a robust feature matching scheme in which features can be matched in $2.3\mu s$. For a typical task involving 150 features per image, this results in a processing time of $500\mu s$ for feature extraction and matching. In order to achieve very fast matching we use simple features based on histograms of pixel intensities and an indexing scheme based on their joint distribution. The features are stored with a novel bit mask representation which requires only 44 bytes of memory per feature and allows computation of a dissimilarity score in 20ns. A training phase gives the patch-based features invariance to small viewpoint variations. Larger viewpoint variations are handled by training entirely independent sets of features from different viewpoints.*

*A complete system is presented where a database of around 13,000 features is used to robustly localise a single planar target in just over a millisecond, including all steps from feature detection to model fitting. The resulting system shows comparable robustness to SIFT [8] and Ferns [14] while using a tiny fraction of the processing time, and in the latter case a fraction of the memory as well.*

## 1. Introduction

Matching the same real world points in different images is a fundamental problem in computer vision, and a vital component of applications such as automated panorama stitching (*e.g.* [2]), image retrieval (*e.g.* [16]) and object localisation (*e.g.* [8]).

Matching schemes must define a measure of similarity between parts of images, which in the ideal case is high if the image locations correspond to the same real-world point and low otherwise. The most basic description of a region of an image is a patch of pixel values. Patch matches can be found by searching for a pair of patches with a high cross-correlation score or a low sum-of-squared-differences (SSD) score. However patch matching with SSD provides no invariance to common image transformations such as viewpoint change, and performing exhaustive patch matching between all possible pairs of patches is infeasible.

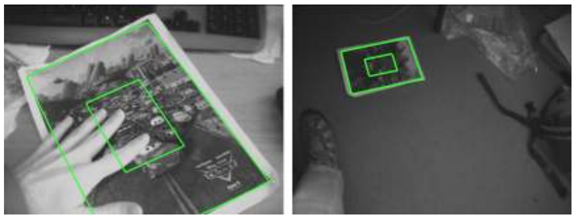
Figure 1. Two frames from a sequence including partial occlusion and significant viewpoint variation. The average total processing time per 640x480 frame for the sequence is 1.37ms using one core of a 2.4GHz processor. Extracting runtime patch descriptors and finding matches in the database accounts for $520\mu s$ of this time.

Moravec proposed an interest point detector [13] to introduce some invariance to translation and hence reduce the number of patch matches to be considered. Interest point detection is now well-established as the first stage of state-of-the art matching schemes. There are many other transformations between the images, such as rotation and scale, which an ideal matching scheme should cope with. There are generally two approaches possible for each category of transformation; either factor out the effect of the transformation, or make the representation of the area of interest invariant to it. Detecting interest points falls into the first category in that it factors out coarse changes in position.

Schmid and Mohr [16] presented the first interest point approach to offer invariance to many image transformations. A number of rotationally invariant features were computed around interest points in images. During matching the same features were computed at multiple scales to give the method invariance to both scale and rotation changes around the interest point.

Instead of computing features invariant to rotation, a *canonical orientation* can be computed from the region around an interest point and used to factor out the effect of rotation. A variety of methods for finding orientation have been proposed including the orientation of the largest eigenvector in Harris [4] corner detection, the maxima in an edge orientation histogram [8] or gradient direction at a very coarse scale [2].

The interest point detection stage can also factor out more than just translation changes. Scale changes can be accounted for by a searching for interest regions over scale space [8, 10]. The space of affine orientation has too many dimensions to be searched directly, so schemes have been proposed to perform local searches for affine orientation starting from scale-space interest regions [11]. Alternatively, interest regions can be found and affine orientation deduced from the shape of the region [9].

Schemes such as those above can factor out large changes due to many common imaging transformations, but differences between matching patches will remain due to errors in the assignment of the canonical parameters and unmodelled distortions. To give robustness to these errors the patches extracted from the canonical frames undergo a further stage of processing. Lowe's SIFT (scale invariant feature transform) method [8] typifies this approach and uses soft binning of edge orientation histograms which vary weakly with the position of edges.

Other systems in this category include GLOH (Gradient Location and Orientation Histogram) [12] and MOPS (Multi-scale Oriented Patches) [2] which extracts patches from a different scale image to the interest region detection. Winder and Brown applied a learning approach to find optimal parameters for these types of descriptor [18]. The CS-LBP descriptor [5] uses a SIFT-style histogram of local information from the canonical patches but the local information used is a binary pattern rather than the local gradient used in SIFT.

All of the above approaches aim to compute a single descriptor for a real-world feature which is as invariant as possible to all likely image transformations. Correspondences between images are determined by extracting descriptors from both images and finding those that are close neighbours in feature space.

An interesting alternative approach recasts the matching problem as one of classification. This approach uses a training stage to train classifiers for the database features, which allows matching to be performed with less expensive computation at run-time than required by descriptor-based methods. Lepetit *et al*. demonstrated real-time matching using randomised trees to classify patches extracted from location, scale and orientation-normalised interest regions [7]. Only around 300 bits are computed from the query images for each interest region to be classified. Later work from Oyuzal *et al*. introduced the Ferns method [14] which improved classification performance to the point where the orientation normalisation of interest regions was no longer necessary. These methods only perform simple computations on the runtime image, however the classifiers need to represent complicated joint distributions for each feature and so a large amount of memory is required. This limits the approach to a few hundred features on standard desktop PCs.

Runtime performance is of key importance for many applications. The template tracking system of Jurie and Dhome [6] performs well but in common with any tracking scheme relies on small frame-to-frame motion and requires another method for initialisation. Recent work on adapting the SIFT and Fern approaches to mobile phones [17] made trade-offs to both approaches to increase speed whilst maintaining usable matching accuracy. Our method is around 4 times faster than these optimised implementations and acheives more robust localisation.

Existing state-of-the-art matching approaches based on descriptor computation or patch classification attempt to match any possible view of a target to a small set of key features. Descriptor-based approaches such as SIFT factor out image transformations with computationally expensive image processing. Classification methods such as Ferns offer reduced runtime computation but have a high memory cost to represent the complex joint distributions involved.

Our method avoids the complexity inherent to matching areas of images subject to large transformations. Instead we employ a training phase to learn independent sets of features for different views of the target, and insert them all into the database for the target. The individual features are only invariant to small changes of viewpoint. This simplifies the matching problem so neither the computationally expensive normalisation over transformations of SIFT-style methods or the complex classifier of the Fern-like approach are required.

As we only require features to be invariant to small viewpoint changes we need far less invariance from our interest point detector than other matching schemes. The FAST-9 (Features from Accelerated Segment Test) detector [15] is a perfect fit for our application as it shows good repeatability over small viewpoint variations and is extremely efficient as it requires no convolutions or searches over scale space.

A potential problem with using features with less invariance than those of other approaches is that more database features will be required to allow robust matching over equivalent ranges of views at runtime. Therefore to make our new approach feasible we require features that have a low memory footprint and which permit rapid computation of a matching score. Our novel bit-mask patch feature fulfils these criteria.

As runtime performance is our primary concern we would like to avoid too much processing on the pixels around the detected interest points. Using pixel patches would be one of the simplest possible matching schemes but SSD-based patch matching would not even provide the small amount of viewpoint invariance we desire. One of the reasons SSD is very sensitive to registration errors is that it assigns equal weight to errors from all the pixels in the patch. Berg and Malik [1] state that registration errors, at least for scale and rotation, will have more effect on samples further from the centre of the patch. The authors reduce the weight of errors in those samples by employing a variable blur which is stronger further from the centre of the patch. We use the idea that not all pixels in a patch are equally important for matching, but further note that the weights which should be assigned to pixels also depend on the individual feature: samples in the centre of large regions of constant intensity will be robust to small variations in viewpoint.

We employ a training phase to learn a model for the range of patches expected for each feature. This model allows runtime matching to use simple pixel patches whilst providing sufficient viewpoint invariance for our framework. For fast localisation the memory and computational cost of matching is reduced by heavily quantising the model to a small binary representation that can be very efficiently matched at runtime.

### 1.1. Our Contributions

- We show fast and robust localisation of a target using simple features which only match under small viewpoint variations.
- A large set of features from different views of a target are combined to allow matching under large transformations.
- We introduce a simple quantised-patch feature with a bit mask representation which enables very fast matching at runtime. The features represent the patch variations observed in a training phase.

## 2. Learning Features for a Target

We use a large set of training images covering the entire range of viewpoints where localisation is required. The set of images could be captured for real, but we instead artificially generate the set by warping a single reference image. Different scales, rotations and affine warps are included in the training set. Additionally random pixel noise and a blur of a small random size are added to each generated view so the trained features have more robustness to poor quality images.

The training views for a target are grouped into several hundred viewpoint bins so that each bin covers a small range of viewpoints. The interest point detector is run on each image in the bin in sequence and patches are extracted from around the detected corners. The interest point locations can be converted to a position in the reference frame as the warp between the reference and training image is known. If the database for the viewpoint already contains a feature nearby the detected point in the new training image, then the patch model for that feature is updated with the new patch. Otherwise a new feature is created and added to the database. When all of the images in a viewpoint bin have been processed we select the $n$ features (typically 50-100) which were most repeatably detected by the FAST detector and quantise their patch models to the binary feature descriptions used at runtime as described in the following section.

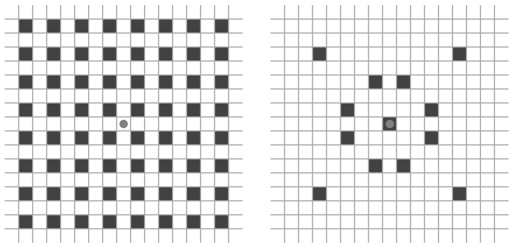
Figure 2. Left: The sparse $8\times 8$ sampling grid used by the features. Right: The 13 samples selected to form the index.

### 2.1. Database Feature Representation

The features in our system are based on an $8 \times 8$ pixel patch extracted from a sparsely sampled grid around an interest point, as shown in Figure 2. The extracted samples are firstly normalised such that they have zero mean and unity standard deviation to give robustness to affine lighting variations. During training we build a model of the feature which consists of 64 independent empirical distributions of normalised intensity, one per pixel of the sampling grid.

This model can be used to calculate the likelihood that a runtime patch is from a trained feature, assuming each pixel is independent. However computing this likelihood estimate would require too much memory and computation time to be used in real-time on a large database of features. Since features only need to match over small viewpoint ranges we are able to heavily quantise the model for a feature and still obtain excellent matching performance.

We quantise the per-pixel distribution in two ways. Firstly the empirical intensity distributions are represented as histograms with 5 intensity bins. Secondly when training is complete we replace the probability in each bin with a single bit which is 1 if pixels rarely fell into the bin (less than 5% of the time). The quantisation is illustrated in Figure 3.

A feature in the database $D$ can be written as:

$$\begin{matrix} D_{0,0} & D_{0,1} & D_{0,2} & D_{0,3} & D_{0,4} \\ D_{1,0} & D_{1,1} & D_{1,2} & D_{1,3} & D_{1,4} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ D_{63,0} & D_{63,1} & D_{63,2} & D_{63,3} & D_{63,4}, \end{matrix} \tag{1}$$

where a row $D_{i,\ldots}$ corresponds to the quantised histogram

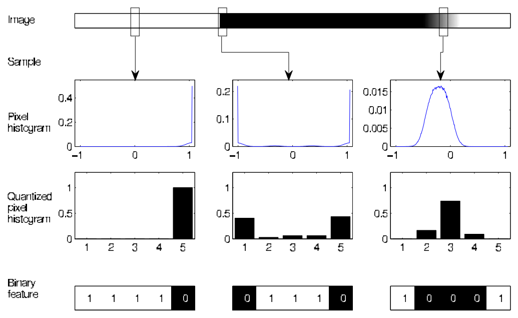

Figure 3. The independent per-pixel empirical distributions are quantised into 5 intensity bins, and then further quantised into a bit mask identifying bins rarely observed during the training phase. This process is shown for: (left) a constant intensity region, (centre) a step change in intensity, (right) an intensity ramp. The data was created by taking the image (top) and adding random blur, noise and translation errors.

for a single pixel of the patch, and

$$D_{i,j} = \begin{cases} 1 & \text{if } \mathrm{P}(B_j < \mathrm{I}(x_i, y_i) < B_{j+1}) < 0.05 \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

where $B_j$ is the minimum intensity value of histogram bin $j$ and $\mathrm{I}(x_i, y_i)$ is the normalised value of pixel $i$.

The resulting descriptor requires 5 bits for each of the 64 samples giving a total of 40 bytes of memory per feature. 4 additional bytes are used to store the position of the feature in the reference image.

## 3. Runtime Matching

After the quantisation to bits the patch descriptions no longer represent probability distributions and so we cannot compute the likelihood of a feature giving rise to a patch. However the bit mask does identify the intensity bins that samples rarely fell into at training time and so good matches should only have a small number of samples which fall into these bins in the runtime patch. Hence we use a count of the number of samples which fall into bins marked with a 1 in the database patch description as our dissimilarity score. The best matching feature in the database is the one that gives the lowest dissimilarity score when compared to the query patch, as that represents the match with fewest "errors" (runtime pixels in unexpected bins). The major advantage of the simple error count measure is that it can be computed with bitwise operations, which allows a large number of potential matches to be scored very quickly.

The bitwise representation of a runtime patch $R$ is slightly different to the database feature of equation 1. It is also represented by a 320-bit value but has exactly 1 bit set for each pixel, corresponding to the intensity bin which the sample from the runtime patch is in:

$$R_{i,j} = \begin{cases} 1 & \text{if } B_j < \mathrm{RP}(x_i, y_i) < B_{j+1} \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

where $\mathrm{RP}(x_i, y_i)$ is the value of pixel $i$ in the normalised runtime patch extracted from around an interest point detected in a runtime image.

With the preceeding definitions of the database and runtime patch representations the dissimilarity score can be simply computed by counting the number of bits where both $D_{i,j}$ and $R_{i,j}$ are equal to 1:

$$e = \sum_{i,j} D_{i,j} \otimes R_{i,j}, \quad (4)$$

where $\otimes$ is a logical AND. Since each row of $R$ always has one single bit set, this can be rewritten as:

$$e = \sum_{i} \left( (D_{i,0} \otimes R_{i,0}) \oplus \ldots \oplus (D_{i,4} \otimes R_{i,4}) \right) \quad (5)$$

where $\oplus$ denotes logical OR. By packing each column of $D$ and $R$ into a 64 bit integer ($\mathbf{D_j}$ and $\mathbf{R_j}$) the necessary logical operations can be performed for all rows in parallel. The dissimilarity score can thus be obtained from a bitcount of a 64-bit integer:

$$e = \text{bitcount}\left( (\mathbf{D_0} \otimes \mathbf{R_0}) \oplus \ldots \oplus (\mathbf{D_4} \otimes \mathbf{R_4}) \right) \quad (6)$$

Computing the error measure therefore requires 5 ANDs, 4 ORs and a bit count of a 64 bit integer. Some architectures (including recent x86 CPUs with SSE4.2) support a single-instruction bitcount. For other architectures, including our test machine, the bitcount can be performed in 16 instructions using an 11 bit lookup table to count chunks of 11 bits at a time. The total time to compute an error measure using the lookup table bitcount is about 20ns.

The first stage of finding matches from a runtime image is to run the FAST-9 interest point detector. As the training phase has selected the most repeatable FAST features from each viewpoint it is not necessary to obtain too many interest points from the input image. We typically find no more than 200 are needed for robust localisation. The $8 \times 8$ patch of Figure 2 is extracted, and the mean and standard deviation of the samples are calculated to enable quantisation into the 320-bits $R_{i,j}$ of equation 3. The dissimilarity score between the patch and each database feature is computed using the fast method of equation 6.

The database feature with the lowest dissimilarity score for a runtime patch is treated as a match if the error count is below a threshold (typically 5). The matches from all the runtime patches can be sorted by error count to order them in terms of quality.

### 3.1. Indexing

The dissimilarity score between a runtime patch and a database feature can be computed very quickly using equation 6, however as we use larger numbers of features than alternative approaches it is desirable to combine the basic method above with an indexing scheme to reduce the number of scores which must be computed and to prevent the search time growing linearly with the database size.

The indexing approach we use is inspired by the Ferns work[14] which uses joint distributions of simple binary tests from training images. Our current implementation uses the 13 samples shown on the right of Figure 2 to compute an index number. The samples have been selected reasonably close to the patch centre as they are expected to be more consistent under rotation and scale, but somewhat spaced apart so that they are reasonably uncorrelated.

Each of the samples selected for the index is quantised to a single bit: 1 if the pixel value is above the mean of the patch and 0 otherwise. The 13 samples are then concatenated to form a 13-bit integer. Thus the index in our current implementation can take values between 0 and 8192. The index value is used to index a lookup table of sets of database features. At runtime the dissimilarity score is only computed against the set of features in the entry of the table with the matching index.

The training phase is used to determine the set of index values which will account for most possible runtime views of a particular feature. Every patch from the training set that contributes to the model for a particular feature also contributes a vote for the index value computed from the patch. After training is complete we select the most-common indices until together the selected set of indices account for at least $80\%$ of the training patches used in building the feature. This set of indices is saved with the feature, and the feature is inserted into all of the corresponding sets of features in the lookup table at runtime.

### 3.2. Improving Robustness to Blur

FAST is not an inherently multi-scale detector and fails to detect good features when the image is significantly blurred. Although our training set includes some random blur so the features are trained to be robust to this we still rely on the repeatability of the detector to find the features in the first place. The few frames where blur is a problem in typical image sequences do not justify switching to a multi-scale detector, so we take a different approach.

To perform detection in blurred images, we create an image pyramid with a factor of 2 in scale between images, and run FAST on each layer of the pyramid. In order to avoid incurring the cost of building the pyramid at each frame, we use a data driven approach to decide when to stop building the pyramid.

Initially features are extracted and matched on the full-sized image. The features are then fed to the next stage of processing, such as estimating the camera pose. If the later stages of processing determine that there are too few good matches, then another set of features are extracted from the next layer of the image pyramid. These are aggregated with the first set of features, but the new features are assumed to have a better score. If again insufficient matches are found, the next layer of the pyramid is used and so on until either enough good matches or a minimum image size has been reached.

We choose a factor of 2 between images in the pyramid, as this allows for a particularly efficient implementation such that around $200\mu$s are required to half-sample a $640 \times 480$ frame. We build a pyramid with a maximum of 3 layers. The resulting system obtains considerable robustness to blur, since the blur in the smallest layer is reduced by a factor of 4. Furthermore, it allows for matches to be made over a greater range of scales as the automatic fallback to sub-sampled images allows matching on frames when the camera is closer to the target than any training images.

## 4. Results and Discussion

In order to validate our method, we apply it to the task of matching points in frames of a video sequence to a known planar object, and finding the corresponding homography. After finding matches the homography is estimated using PROSAC [3] and refined using the inliers. The inlier set is reestimated and refined for several iterations. The resulting homography allows us to determine which points were matched correctly.

The database for the frames shown in Figure 1 was generated from a training set of 21672 images, generated by warping a single source image of the target. 7 different scale ranges and 36 different camera axis rotation ranges were used, giving a total of 252 viewpoint bins. Each bin covers a reduction in scale by a factor of 0.8, 10 degrees of camera axis rotation, and out-of-plane viewpoints in all directions of up to 30 degrees. We extract around 50 features from each viewpoint bin (more from larger scale images), giving a total of 13372 features in the database.

### 4.1. Validating the Bit Count Dissimilarity Score

Two short video sequences of the planar target of Figure 1 were captured using a cheap VGA webcam. The first sequence was captured from viewpoints which were known to have been covered by our training phase whereas the second sequence was viewed with a larger out-of-plane rotation, known to be outside the range of training. The database features were trained from the source image, whereas the test sequences were poor-quality webcam images of a printed version of the file. Thus both sequences test the method's

| | |
|---|---|
| FAST interest point detection | 0.55ms |
| Building query bit masks | 0.12ms |
| Matching into database | 0.35ms |
| Robust pose estimation | 0.1ms |
| Total frame time | 1.12ms |

Table 1. Timings for the stages of our approach on a dataset with images taken from within the range of trained viewpoints.

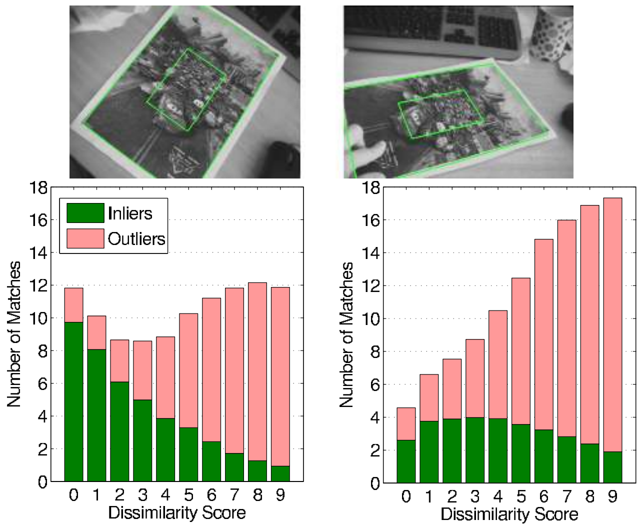

Figure 4. The bit error count provides a reasonable way to determine good matches. Left: matches from viewpoints contained in training set. Right: matches on viewpoints from outside training set.

robustness to different imaging devices.

Matching on the first test sequence was very good, correctly localising the target in all 754 frames of the test sequence. There was little blur in the sequence so the full frame provided enough matches in all but 7 frames of the sequence, when the half-sampled image fallback was used to obtain enough matches for a confident pose estimate. The average total frame time on the sequence was 1.12ms on a 2.4GHz processor. The time attributed to each stage of the process is shown in Table 1.

Somewhat surprisingly our method also performed reasonably well on the second sequence, even though it was known the frames were taken from views that were not covered by our training set. On this sequence the target was localised in 635 frames of the 675 in the sequence (94%). As expected the pose estimate using only the full-frame image was generally less confident so the fallbacks to sub-sampled images were used more often: 377 frames used the half-image and 63 also used the quarter-scale image. Because of this additional workload the per-frame average time increased to 1.52ms.

The matching performance on these test sequences suggests that the bit count dissimilarity score provides a reasonable way of scoring matches. To confirm this we computed the average number of inlier and outlier matches over all of the frames in the two sequences, and plotted these against the dissimilarity score obtained for the match in Figure 4. For the sequence on the left where the viewpoints are included in the training set many good matches are found in each frame, with on average 9.7 zero-error inliers obtained. The inlier percentage for matches with low dissimilarity scores is also good at over 82% in the zero error case. The result that both the number of inliers and the inlier fraction drop off with increasing dissimilarity score demonstrates that the simple bit error count is a reasonable measure of the quality of a match. The figure provides strong support for a PROSAC-like robust estimation procedure once the matches have been sorted by dissimilarity score as the low error matches are very likely to be correct.

Even when the viewpoint of the query image is outside the range for which features have been trained, as in the data on the right of Figure 4, the dissimilarity score still provides a reasonable way to sort the matches, as the inlier fraction can be seen to drop off with increasing dissimilarity. The inlier rate of the first matches when sorted by dissimilarity score is still sufficient in most frames to obtain a pose with a robust estimation stage such as PROSAC.

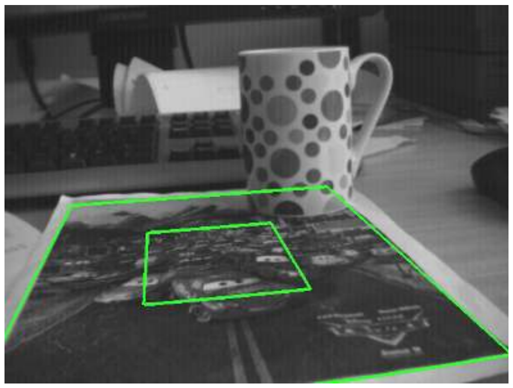

Figure 5. Increasing the range of viewpoint bins in the training set allows more viewpoint invariance to be added in a straightforward manner.

## 4.2. Controllable Viewpoint Invariance

As our framework uses independent features for different viewpoint bins it is possible to trade-off between robustness to viewpoint variations and computation required for localisation by simply adding or removing more bins.

For applications where viewpoints are restricted (for example if the camera has a roughly constant orientation) the number of database features can be drastically reduced leading to even higher performance. Alternatively if more computational power is available it is possible to increase the

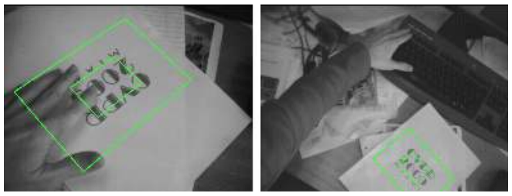

Figure 6. Targets lacking in texture can be localised successfully over a large viewpoint range and in the presence of background clutter and partial occlusion.

| | Poster | Logo |
|---|---|---|
| Our Method | 625 | 512 |
| Our Method, 5000 iterations | 632 | 517 |
| SIFT, Distance | 590 | 517 |
| SIFT, Ratio | 629 | 517 |
| SIFT, Ratio, 5000 iterations | 630 | 517 |

Table 2. Comparison of the number of frames where the target was localised in two sequences. The Poster dataset is as in Figure 1 and consists of 637 frames and the Logo dataset is shown in 6 and has 517 frames.

viewpoint invariance of the method by adding features from more viewpoint bins. Figure 5 shows the same database as before, with some additional viewpoint bins trained with more extreme out-of-plane rotations. However the number of bins required to cover the full space of affine variations is large. We find in practice a single top-down-centred bin at each scale and rotation which includes small affine changes results in a runtime system which is able to localise the target successfully in a range of viewpoints comparable to other methods without the specifically trained extreme out-of-plane features in the database shown in Figure 5.

## 4.3. Targets Lacking Detailed Texture

As we only require a small number of interest points to be found from each viewpoint our method also works well on scenes lacking detailed texture, as shown in Figure 6. The database for this target contained 10800 features. Our method found a pose for the object in 512 of the 517 frames in the sequence (99%). The average frame time was 1.85ms, which is increased from the other sequences as the lack of texture means many features fall in the same index bins resulting in more comparisons at runtime.

## 4.4. Comparisons with Other Methods

We compared the performance of our method against the widely used SIFT technique on the sequences shown in Figures 1 and 6. SIFT keypoints were extracted from every frame of the two sequences using David Lowe's binary. Keypoints were also extracted from the same reference image used to generate the training sets for our method. Around 3500 SIFT keypoints are found in the poster image, and 500 for the logo.

We exhaustively search through all of the reference SIFT descriptors to find the nearest neighbour to every descriptor extracted from each frame of the test sequence. We do not use any threshold to reject mismatches but instead sort the matches and apply PROSAC which automatically favours the best-scoring matches. We use two different scores to sort the matches; one is the distance to the nearest reference keypoint in SIFT space, and the second is the ratio of the distances to the two nearest neighbours, as commonly used in practice. We firstly allow 150 prosac iterations to match the parameters used by our runtime method (which uses a maximum of 50 PROSAC iterations on each image from the pyramid). Secondly we run both SIFT and our method with 5000 PROSAC iterations to see if this is able to correctly localise the target in more frames.

The number of frames matched successfully for each sequence is shown in Table 2. The results show that we manage to successfully localise the targets in a similar number of frames to SIFT. They also validate the ratio of distance measure as being the best indication of the quality of a SIFT match. The parameters in our method are tuned towards speed rather than accuracy but we still demonstrate excellent performance on these representative real-world data sets. Interestingly when we use more iterations of PROSAC on our approach we manage to localise more of the frames, and in the Poster sequence we localise more frames than the exhaustive SIFT implementation which is a promising result. The improvement in performance with more PROSAC iterations suggests the error count dissimilarity score alone may not be good enough to sort matches in challenging images. We could apply additional higher level checks (for example, viewpoint consistency) to identify the matches which are likely to be the best so they can be promoted to the start of the list for the PROSAC procedure.

The SIFT binary requires around 1 second per frame to extract the keypoints. N-to-N matching would not be used in practice with large databases, but it is clear the amount of computation required both for keypoint extraction and matching is far lower with our approach and we expect even with optimised SIFT code our method would be hundreds of times faster in practice. Even so it should be noted that we require a training phase of around 1 hour per target to achieve the high performance at runtime so our method is not suitable for all applications.

The Ferns approach is similarly targeted at real-time localisation applications. The authors report a total per-frame time of around 20ms on similar hardware to our testing PC. Our approach is around ten times faster, and as we do not represent joint distributions we require around 100 times less memory than used by a Fern implementation with typical parameters.

Wagner *et al*. report an average total frame time of around 5ms on $320 \times 240$ sequences for their speed-optimised SIFT and Fern implementations[17]. The authors use a test sequence with the same target as Figure 1 and report both methods localise the target in around 96% of the frames. Our method applied to the same sequence localises 99.6% of the frames with an average frame time of 1.04ms, so is over 4 times faster than both approaches in [17] and provides more robust localisation on this sequence.

## 5. Conclusions and Future Work

We have proposed a novel solution to the feature matching problem using independent features from many different viewpoint bins which are only required to be invariant to small viewpoint changes. This means simple features can be used with very low memory and computational requirements but can still provide matching performance comparable to other state-of-art schemes requiring many orders of magnitude more computation at runtime.

Our novel binary feature representation for a pixel patch is trained offline to give the local invariance required within the viewpoint bin. The binary feature requires just 44 bytes of memory, and can compute a dissimilarity score against a query patch in just 20ns. This allows our entire target localisation (including feature detection, query patch extraction, matching, and robust pose estimation) to run in around 1.5ms per frame on average, ten times faster than the Ferns approach and hundreds of times faster than approaches based on computationally expensive image processing such as standard SIFT matching.

The parameter space of our features has not been explored so we would like to apply a learning approach to discover the optimal number of quantisation bins and the best number of samples and their layout for the binary features.

The FAST detector offers good repeatability with camera-axis rotation and so we could try and normalise for rotation at training time. As we currently use 36 separate viewpoint bins to cover the range of camera rotation this has the potential to hugely reduce the number of features we require for a single target.

## 6. Acknowledgements

This research is supported by the Boeing Company.

## References

[1] A. C. Berg and J. Malik. Geometric blur for template matching. In *Computer Vision and Pattern Recognition*, volume 1, pages 607–614, 2001. 3

[2] M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 1:510–517 vol. 1, June 2005. 1, 2

[3] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In C. Schmid, S. Soatto, and C. Tomasi, editors, *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 220–226, Los Alamitos, USA, June 2005. IEEE Computer Society. 5

[4] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, pages 147–151, 1988. 2

[5] M. Heikkilä, M. Pietikäinen, and C. Schmid. Description of interest regions with local binary patterns. *Pattern Recogn.*, 42(3):425–436, 2009. 2

[6] F. Jurie and M. Dhome. Hyperplane approximation for template matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):996–1000, Jul 2002. 2

[7] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *18th IEEE Conference on Computer Vision and Pattern Recognition*, San Deigo, California, USA, June 2005. Springer. 2

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 2:91–110, 2004. 1, 2

[9] J. Matas, O. Chum, M. Urbana, and T. Pajdlaa. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, Sept. 2004. 2

[10] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *8th IEEE International Conference on Computer Vision*, volume 1, pages 525–531, Vancouver, Canada, 2001. Springer. 2

[11] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *Proceedings of the 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002. Copenhagen. 2

[12] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(10):1615–1630, 2005. 2

[13] H. Moravec. Rover visual obstacle avoidance. In *proceedings of the seventh International Joint Conference on Artificial Intelligence*, pages 785–790, August 1981. 1

[14] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 1:1–8, June 2007. 1, 2, 5

[15] E. Rosten and T. Drummond. Machine learning for high speed corner detection. In *9th Euproean Conference on Computer Vision*, volume 1, pages 430–443. Springer, Apr. 2006. 2

[16] C. Schmid and R. Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997. 1

[17] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Pose tracking from natural features on mobile phones. In *Proc. ISMAR 2008*, Cambridge, UK, Sept. 15–18 2008. 2, 8

[18] S. A. Winder and M. Brown. Learning local image descriptors. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007. 2