Multiple Target Localisation at over 100 FPS

Simon Taylor
http://mi.eng.cam.ac.uk/~sjt59
Tom Drummond
http://mi.eng.cam.ac.uk/~twd20

Department of Engineering
University of Cambridge
Cambridge, UK

Abstract

This paper presents a method for fast feature-based matching which enables 7 independent targets to be localised in a video sequence with an average total processing time of 7.46ms per frame. We extend recent work [14] on fast matching using Histogrammed Intensity Patches (HIPs) by adding a rotation invariant framework and a tree-based lookup scheme. Compared to state-of-the-art fast localisation schemes [15] we achieve better matching robustness in under a quarter of the computation time and requiring 5-10 times less memory.

1 Introduction

Finding points in different views of a scene which correspond to the same real world locations 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.* [13]) and object localisation (*e.g.* [7]).

A common theme in many successful approaches to these problems is the extraction of a set of local features from images to be matched. An overall match between two images can then be established by combining information from many local feature correspondences. The use of information from many local matches adds redundancy and allows the methods to cope with partial occlusion and some incorrect correspondences.

The first stage of all state-of-the-art matching schemes is to apply interest region detection to factor out common imaging transformations. The Harris corner detector [4] has been used frequently, but modern methods commonly use more expensive searches for scale-space interest regions such as the DoG detector [7]. Finding affine-invariant interest regions has also been extensively studied [8, 9]. A canonical orientation can be assigned to an interest region, for example by considering the blurred gradient at the centre of the region [2].

The most basic representation of the interest region is obtained by extracting a pixel patch from the canonical frame that has been assigned. Simple patch-matching schemes such as Normalised Cross Correlation (NCC) or Sum-of-Squared Differences (SSD) do not perform well when subject to the small registration errors introduced by interest region detectors, and hence a more complicated matching scheme is usually employed. Two broad approaches have been studied in the literature. The first class of methods perform further processing on the extracted patches to compute a feature descriptor vector which is ideally equal for different views of the same feature. The second class of methods treats the matching problem

BMVC 2009 doi:10.5244/C.23.58

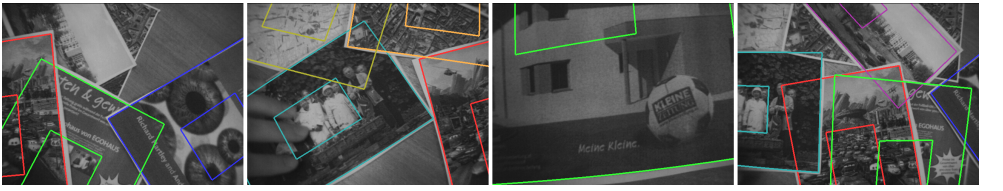

Figure 1: Four frames from a sequence demonstrating independent multiple object localisation. No frame-to-frame tracking is performed; the objects are localised in each frame. The mean total computation time per frame is 7.46ms using a single core of a 2.4GHz CPU.

as one of classification, and can obtain matches with very little computation on the input patches after classifiers for each database feature have been learnt in an offline training stage.

David Lowe's SIFT method [7] is one of the most common descriptor-based approaches. SIFT uses blurring and soft-bin histogramming of local gradients to extract a descriptor vector which is robust to the errors introduced by interest region detection and orientation assignment. Many other approaches to transforming an image patch into a feature vector have been proposed, such as GLOH [10], MOPS [2], and CS-LBP [5]. Winder and Brown applied a learning approach to find optimal parameters for these types of descriptor [16].

Lepetit *et al*. [6] demonstrated the viability of the alternative matching-by-classification approach by showing that an offline training phase could be used to train randomised tree classifiers for features. The Ferns method [11] uses a different classifier with improved performance. Both of these methods only require a small number of simple pixel tests on the runtime images to classify features, and hence require very little computation. However the classifiers represent joint distributions of the tests and so have large memory requirements.

Both the SIFT and Ferns approaches as originally presented require too much memory and computation to be suitable for real-time applications on small devices such as mobile phones. Recent work by Wagner *et al*. [15] adapted both approaches to make them suitable for low-powered platforms. A key change to both methods was replacing the expensive scale-space search for interest regions with the very efficient fixed-scale FAST-9 detector [12], and instead achieving scale invariance at runtime by ensuring the database contains features from multiple scales. Their optimised methods were both able to localise a planar target in a 320×240 image in a total frame time of around 5ms on a desktop PC. The technique we propose achieves more robust localisation on the same test sequences and reduces both the total frame time and memory requirements by a factor of more than 4.

Our approach is based on simple pixel patches extracted from around interest points. Although SSD-based matching is not robust when subject to small registration errors, registration errors do not affect all pixels equally; samples from the interior of large regions of solid colour in a patch are more robust to registration errors. We employ a training phase to learn a model for the range of patches expected for each feature using independent per-pixel distributions, which we refer to as a Histogrammed Intensity Patch (HIP). This model allows runtime matching to use simple pixel patches whilst providing sufficient viewpoint invariance to handle registration errors from interest point detection. The histograms are quantised to give a small binary representation that can be very efficiently matched at runtime.

In previous work [14] we introduced the approach and used the efficient FAST-9 detector to factor out translation changes. We trained independent sets of features from many different viewpoint bins covering scale, rotation and affine variations, which resulted in large

databases of around 13,000 features for a single target. Despite the large database size, a simple indexing scheme combined with the binary representation's efficient matching score enabled localisation of a target in a total frame time of around 1.5 milliseconds.

This paper presents a number of significant improvements to the approach. Firstly canonical orientation computation is added to the interest point detection stage, allowing the number of features for a target to be reduced by a factor of around 15. A novel two-pass approach to training accounts for errors related to interest point detection and orientation assignment, and allows us to choose efficient methods for those stages without sacrificing robustness of the overall system. A tree-based matching scheme is introduced to exploit common information in different features and prevent the need for an exhaustive comparison against all database features. Finally a framework for rapid independent multiple-target localisation using HIPs is presented.

2 Training Features for a Target

As in the Ferns [11] approach we utilise a training phase to reduce the amount of computation required at runtime. We use a training set of views of the entire target covering the range of viewpoints for which localisation is desired. The set of views is artificially generated by warping a single reference image.

For maximum runtime performance we avoid scale or affine-invariant interest region detectors and instead train independent sets of features for different viewpoint "bins", each of which cover a small range of scale and affine viewpoint parameters. The experiments in this paper use 9 scale bins in total, 3 per octave. In practice we only use one range of affine parameters representing viewpoints centred on a direct view of the target but including out-of-plane rotations of up to 40 degrees in all directions. This single set of affine parameters enables reasonable matching beyond this range, but additional extreme affine viewpoint bins could be added if required.

Around 1000 images are generated for each viewpoint bin. Each image is generated by warping the reference image with a random camera-axis rotation and randomly chosen scale and affine parameters from within the range of the viewpoint bin. Additionally a small random gaussian blur and pixel noise is added so the training set more accurately represents the poor quality images likely at runtime. Our current implementation takes around 20 minutes to generate all the training images for a typically-sized target, and uses large kernels convolved with the reference image to avoid aliasing artifacts.

The images in the training set are similar to the frames we expect at runtime, although as they are warped views of the entire target they are often larger than standard camera resolutions. To ensure all possible camera views contain sufficient features for localisation we split the viewpoints into 200×200 pixel *regions* defined in a *viewpoint reference frame* which is taken as an unrotated view of the target from the centre of the viewpoint bin.

A two-stage training approach is used to identify repeatable features in each region and build feature models for them. The first stage is to run interest point detection and orientation assignment on all of the training images. The position and orientation of each detection and the appearance of the surrounding image region is stored in a structure termed a *subfeature*. The second stage then clusters subfeatures based on position and orientation to identify repeatable features, and builds a Histogrammed Intensity Patch model for each feature by combining the appearance information from the subfeatures in each cluster.

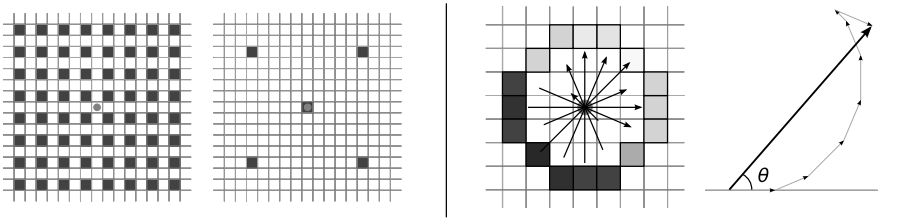

Figure 2: Left: The 8×8 sample grid used for the HIPs and the 5 sample locations selected for indexing, relative to the FAST-9 interest point (shown by the grey circle). Right: The orientation assignment scheme uses a sum of the gradients between opposite pixels in the 16-pixel FAST ring.

2.1 Selecting Repeatable Feature Positions and Orientations

Runtime performance considerations led us to select FAST-9 [12] as the interest point detector. Typical approaches to assigning orientation require computationally expensive blurring [2] or histogramming [7] and would add significant computation to the runtime processing. Instead we simply sum gradients computed between opposite pixels in the 16-pixel ring used in FAST corner detection, as shown in Figure 2. The directions are fixed so the x and y components of the orientation vector can be computed very quickly from weighted sums of the 8 pixel differences.

We run FAST-9 on each training image within a viewpoint bin and represent the 35 highest-scoring corners from each 200×200 region with subfeatures. Proportionally fewer subfeatures are extracted from smaller regions at the edges of the viewpoint reference frame. For smaller scale viewpoint bins where the entire target is under 200×200 pixels a 35 corner minimum is enforced which effectively increases the feature density for these smaller targets. The orientation measure of Figure 2 is also computed at each detected corner. The position (x_r, y_r) and orientation θ_r of the subfeature in the coordinate system of the viewpoint reference frame can be computed as the warp used to generate the training image is known.

The appearance of the subfeature is represented by a sparsely-sampled quantised patch. We use a square 8×8 sampling grid centred on the interest point, with a 1-pixel gap between samples, as shown in Figure 2. Before sampling the pixel values the sampling grid is first rotated so that it is aligned with the detected orientation of the subfeature. The 64 samples are then extracted with bilinear interpolation, normalised for mean and standard deviation to give lighting invariance, and quantised into 5 intensity bins. The 5-bit index value explained in Section 3.2 is also computed and stored in the subfeature.

The most repeatable feature positions and orientations for a viewpoint bin appear as dense clusters of subfeatures in the (x_r, y_r, θ_r) space when all the training images in the bin have been processed. Every subfeature is considered as the potential centre of a HIP feature, and the set of other subfeatures (from other training images) that lie within a certain distance of the centre is found. We manually decide the allowable distance; in this paper we allow 2 pixels of localisation error and 10 degrees of orientation error. Sets of subfeatures given these allowed distances share enough similarity in appearance to be represented by a single HIP feature in a target database. The largest set of subfeatures represents the most repeatably-detected feature, and will be the first feature we select to add to the database. Sets continue

to be selected in a greedy manner, disregarding any which overlap already added HIPs. We continue adding features until the average number of subfeatures per training image region which are represented in the database has reached a specified fraction of the number of subfeatures detected. We set this parameter to 0.5 in our experiments. This criterion will naturally compensate for inaccuracies in interest point detection and orientation assignment by adding multiple database features to represent a single cluster if the errors are too large. Hence our use of an inexpensive and inaccurate orientation assignment scheme may result in more features being added to the database but should not cause a major degradation in matching robustness. The trade-off between robustness of matching and database size can be made by adjusting the parameters for desired corner density and desired fraction of corners in the database.

2.2 Creating HIPs from Subfeature Sets

After a particular set of subfeatures has been selected for addition to the database, quantised patches from the subfeatures are combined to give the Histogrammed Intensity Patch representation for the feature. The HIP model contains 64 independent histograms of 5 quantised intensity levels; one histogram for each sample of the quantised patches. The histograms are empirical distributions of the quantised intensity in a particular sample across all the subfeatures in the set. Thus samples which have a consistent quantised level in all of the constituent subfeatures will have sharply peaked histograms in the HIP. To save on memory and computation required for matching we quantise the histograms to a binary representation. Histogram bins with probabilities less than 5% are *rare bins* and represented by a 1, and other bins by a 0. A single HIP requires 5 bits for each of 64 samples, a total of 40 bytes. We can refer to each bit of a database HIP D as $D_{i,j}$ where $i \in \{0,\ldots,63\}$ is the sample number and $j \in \{0,\ldots,4\}$ is the quantised intensity level.

The process of extracting subfeatures from training images, finding the largest subfeature sets and building the HIP representations takes less than 5 minutes on a desktop machine for a typical target with around 10000 images in the training set.

3 Runtime Matching

We use a fixed-scale detector to avoid a dense scale space search at runtime but we still find it useful to build a sparse image pyramid by half-sampling the input image twice to obtain half and quarter-scale images. As well as having significantly reduced blur, which improves repeatability of the fixed-scale FAST detector, these sub-sampled images also enable matching over a wider range of scales as features can still be detected in the reduced images if the target scale in the input image is larger than the trained range. Using half-sampling by averaging 4 pixels for each 1 of the reduced image permits a very efficient SIMD implementation which produces both reduced images from a 640×480 frame in under 0.1ms.

The runtime images are treated similarly to those from the training set: FAST-9 is used to extract the highest scoring corners from the images, the orientation assignment scheme of Figure 2 is employed and a patch is extracted from the rotated sparse sampling grid and normalised for mean and standard deviation. We use bilinear interpolation with precomputed pixel positions and weights in 2 degree increments so the additional cost of rotation normalisation is minimal, and find around 150 corners from the full scale image and 75 from each of the reduced-scale is sufficient for excellent matching robustness.

For efficient matching the normalised patch is converted to a binary representation R. This is slightly different to the database feature D as it represents just a single patch. Like D, we use a 320-bit value but R has exactly one bit set for each pixel, corresponding to the intensity bin for each sample in the patch:

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

where $\text{RP}(x_i, y_i)$ is the value of sample i in the normalised runtime patch, and B_j is the minimum normalised intensity value of histogram bin j.

The dissimilarity score we use to rank correspondences is computed in the same way as our earlier system [14]. In brief; although the binary HIP models do not allow a true likelihood computation as they are not correctly normalised, an approximation which is sufficient to classify matches can be obtained from a count of the number of samples in the runtime patch which fall into the rare bins in a HIP. This can be computed with bitwise operations on the binary representations of D and R - the error count is simply the number of bits where both $D_{i,j}$ and $R_{i,j}$ are equal to 1. By packing the bits corresponding to the same intensity levels into 64-bit integers $\mathbf{D_j}$ and $\mathbf{R_j}$ and using the fact that only one of the $R_{i,j}$ bits will be set for each sample i, the error count can be obtained very efficiently by a bit-count on a 64-bit integer:

$$e = \text{bitcount}\left((\mathbf{D_0} \otimes \mathbf{R_0}) \oplus \ldots \oplus (\mathbf{D_4} \otimes \mathbf{R_4})\right) \quad (2)$$

where $\otimes$ denotes bitwise AND and $\oplus$ denotes bitwise OR. Currently the parallel bit-count method from [1] is used which is slightly faster than the 11-bit lookup table used in [14].

We use two thresholds for determining matches. Matches with $e \leq 2$ are treated as primary matches, whilst those where $2 < e \leq 4$ are secondary matches. As this is a classification-based approach it is possible for a single runtime patch to generate more than one match. This is an advantage as in the case where real world features look genuinely similar we would like to add matches reflecting all possibilities to the set used for robust estimation. Descriptor-based approaches usually only treat the nearest neighbour as a match.

3.1 Tree-based Search

We improve the scalability of our approach and avoid the need to compare every runtime match against every database feature by making use of similarities between HIPs. Two similar-looking features in the database are likely to share many rare bins (1 bits) in their HIP representations. A lower bound on the number of errors between a runtime patch and the two features can be obtained by computing the error score between the patch and the ANDed binary representations of the two features. If this error score is over the threshold for matching then matches to both features can be rejected with only one test.

The use of ANDed bitmasks to reject entire sets of features can be used to build a binary tree. Initially the number of common bits between all pairs of HIPs is computed. The pair with the greatest overlap is converted into a parent feature containing the ANDed masks. The parent is added to the set of root features and the two original features become child nodes of the parent. The process of combining the root features with the most overlap is repeated until the root features which remain do not share any 1 bits in common. Our implementation builds a tree for a typically-sized database of 700 features in under a second by maintaining a sorted list of the pairs of HIPs with the greatest overlap, which only needs to be sparsely updated when a pair is combined.

The tree-based search enables all matches under the threshold to be retrieved from the entire database without requiring an exhaustive comparison. As we use a binary tree the number of additional parent features is almost exactly equal to the number of original database features, so the method requires around twice the memory of a linear search.

3.2 Indexing

Our previous work [14] used a 13-bit index to reduce the exhaustive search. The combination of smaller rotationally-invariant databases and the novel tree-based lookup method make a full search a real possibility. However if matching speed is of key importance an index can be used to further reduce the matching time at the cost of more memory usage. We use a 5-bit index in this paper.

The 5 samples shown in Figure 2 are used to compute an index number. The samples selected for the index are quantised to a single bit, set to 1 if the pixel value is above the mean of the patch, and concatenated to form a 5-bit integer. The value is used to index a lookup table of sets of HIPs, and the error score is only computed against the HIPs in the entry of the table with the matching index. The features in each index bin can also be grouped using the binary tree approach of section 3.1 to further reduce the number of comparisons required at runtime.

The training phase is used to decide the index bins that a feature should appear in. Every subfeature used to build a HIP model also contains the index value computed from the training image it appeared in. The most common indices for a particular HIP are selected until at least 80% of its constituent subfeatures have their index value included in the set. As an index bin only contains a subset of the database it is possible that matches will be missed when the index scheme is employed.

3.3 Robust Multiple Target Pose Estimation

For the single targets used in [14] it was sufficient to sort all of the matches by error score and apply a standard robust estimation framework such as PROSAC [3]. This approach does not scale well to multiple targets as it would require a separate set of PROSAC iterations for each target in the database. We make use of the target, scale bin, and reference orientation associated with each HIP to bin matches into coarse viewpoints which they support. The viewpoints with the most primary matches are considered first. We always choose the top 5 viewpoints for further consideration, and additionally the top two for each target providing they have enough matches to allow a pose to be estimated.

All matches from the viewpoint being considered (and those from neighbouring scale and orientation views) are added to a set of potential matches. We then apply viewpoint consistency constraints to identify the primary match within this set with the largest number of consistent matches. The primary match is consistent with another match if the distance between the matches d_m is close to the expected distance d_e given the scale bin of the primary match: $0.4d_e < d_m < 1.5d_e$. The direction of the vector between the matches must also be consistent with the reference rotation of the primary patch to within 30 degrees. We allow quite large errors as often features correctly match outside of their viewpoint range, and also the expected values assume a straight-on view of the target which may not be true of the input frame. If the biggest consistent set within a viewpoint has more than 6 matches we apply PROSAC to obtain a candidate homography. This is optimised using all of the matches in the viewpoint bin, and if a significant number of inliers are found it is further optimised using

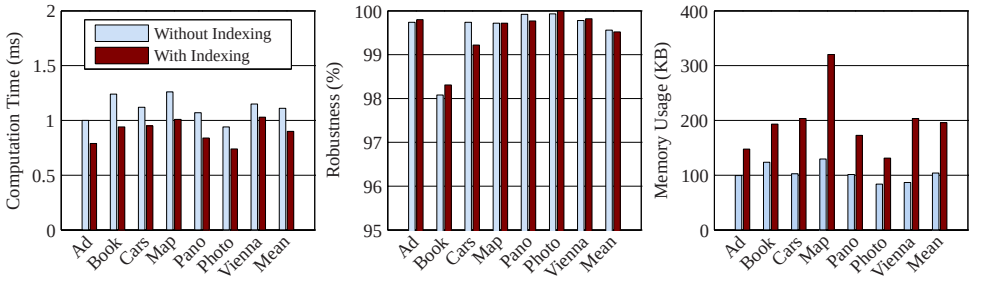

Figure 3: Results on the 7 data sets from [15]. The computation time graph shows the total processing time per frame on a 2.4GHz CPU. Robustness is measured as the percentage of frames where a pose with more than 10 inliers was found.

the entire set of matches. If a pose for a target with a large inlier set is found we remove any matches from runtime patches explained by the homography and repeat the viewpoint based voting and consistency checks with those that remain. This enables all targets present in the scene to be matched, but stops quickly as soon as the remaining matches do not indicate the presence of other targets. The use of high-level consistency checks also ensures the expensive step of checking candidate homographies against all potential matches only occurs when we are reasonably confident the target is visible in the image.

4 Results

4.1 Comparison to Wagner *et al.* [15]

HIP databases were trained for each of the 7 targets used by Wagner *et al.* for the evaluation of their cut-down SIFT and Ferns implementations optimised for small devices. The resolution of the sequences is 320×240. The results we achieve with and without indexing (both methods using the tree-based lookup) are plotted in Figure 3.

The average frame time reported by Wagner *et al.* on these sequences is close to 5ms for both Ferns and SIFT approaches when run on a 2GHz CPU. Even accounting for the slightly increased clock speed of our test machine (2.4GHz) our method is over 3.5 times faster without indexing, and more than 4.5 times faster when using a 5 bit index. The number of frames localised is improved from around 96% reported in [15] to over 99.5% with our HIPs approach. Indexing does not seem to make a significant difference to localisation robustness on these datasets. The implementations of [15] used around 1MB of memory per target for the databases and associated indexing structures. Without indexing our approach offers a memory saving of a factor of around 10. With the 5-bit index the memory requirement of our method approximately doubles, so still offers a factor of 5 reduction over the cut-down Ferns and SIFT implementations.

4.2 Performance With Multiple Targets

To investigate the scalability of our method as the number of database features and targets is increased we ran our runtime matching implementation on the 640×480 video sequence shown in Figure 1 which includes all 7 of the targets. We started with only the features for

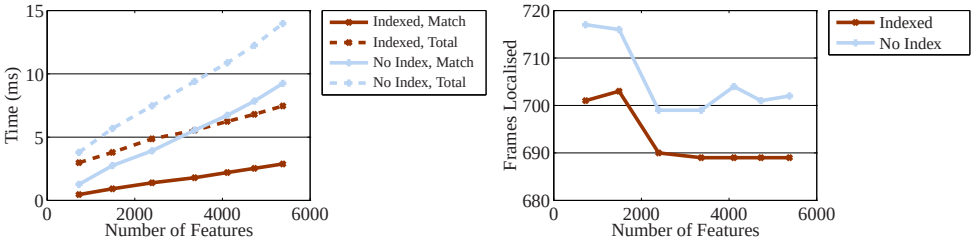

Figure 4: Scalability results on the sequence of Figure 1. Left: Matching and Total Frame Times. Right: Number of frames where "Ad" target is localised.

the "Ad" target in the database, and added the other targets one at a time. The total frame time and the time taken in the match lookup stage alone were recorded. Another parameter of interest was how the robustness of the matching is affected by including more targets in the database, so we also recorded the number of frames in which the "Ad" target was localised. The results were calculated both with and without indexing and are shown in Figure 4. In these tests we increased the number of corners extracted at runtime to 400 from the full frame and 200 from each of the sub-sampled images, as we found the single corner threshold for the entire frame caused the lower-contrast targets to have too few features detected in some frames. Enforcing a spread of corners in the runtime frame could be a better solution and should work with fewer runtime corners, giving a corresponding reduction in matching time.

The speed results appear to show the matching time increasing almost linearly despite the tree-based lookup. It could be that even with all 7 targets in the database there are not enough overlapping features to fully exploit the potential for log(N) scaling of the binary tree. It would be interesting to investigate the behaviour as the database is increased to a much larger number of features. The index clearly helps to reduce the matching time, and potentially an index with more bits could be used if matching time is of utmost importance.

The right side of the figure shows there is a slight performance penalty when the database contains multiple targets. However after a few targets have been added to the database, the addition of further ones does not make performance significantly worse. As expected, the approximation introduced by the indexing scheme leads to some potential matches not being identified, and results in fewer frames being successfully localised. However the penalty is minimal considering the speed increase obtained by using an index.

5 Conclusions and Future Work

We have proposed some significant improvements to our earlier matching scheme based on Histogrammed Intensity Patches. A simple orientation computation and a training framework which is able to naturally deal with inaccuracies in orientation assignment has enabled database sizes to be significantly reduced. A novel binary tree lookup scheme allows an exact search for low-threshold matches without requiring exhaustive comparison. Finally viewpoint consistency constraints have been employed to implement a scalable multiple target localisation framework. Comparison with state-of-the-art fast localisation schemes has demonstrated that our method offers faster performance and lower memory usage whilst also successfully localising the target in more frames of the test sequences.

Future research will investigate reducing training time, learning optimal values for various parameters in the method (such as sample layout, number of samples and quantisation levels), and scalability tests with far larger image databases. We also propose investigating using image measurements other than intensity in the same binary-histogrammed framework; for example samples could also include a histogram of quantised gradient directions or quantised colour. The resulting matching scheme would then naturally give more weight to any of the parameters which were found to be consistent for a particular sample in a feature.

6 Acknowledgements

This research is supported by the Boeing Company.

References

[1] Software optimization guide for AMD64 processors. URL http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/25112.PDF.

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

[3] Ondřej Chum and Jiří Matas. Matching with PROSAC - progressive sample consensus. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 220–226, 2005.

[4] C Harris and M Stephens. A combined corner and edge detector. In *Proc. of the 4th ALVEY Vision Conference*, pages 147–151, 1988.

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

[6] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1465–1479, Sept. 2006.

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

[8] 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, September 2004.

[9] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *Proc. 7th European Conference on Computer Vision, Copenhagen, Denmark*, pages 128–142. Springer, 2002.

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

[11] M. Ozuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2007.

[12] Edward Rosten and Tom Drummond. Machine learning for high speed corner detection. In *9th Euproean Conference on Computer Vision*, volume 1, pages 430–443. Springer, April 2006.

[13] Cordelia Schmid and Roger Mohr. Local greyvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:530–535, 1997.

[14] Simon Taylor, Edward Rosten, and Tom Drummond. Robust feature matching in 2.3μs. In *IEEE CVPR Workshop on Feature Detectors and Descriptors: The State Of The Art and Beyond*, June 2009.

[15] Daniel Wagner, Gerhard Reitmayr, Alessandro Mulloni, Tom Drummond, and Dieter Schmalstieg. Pose tracking from natural features on mobile phones. In *Proc. ISMAR 2008*, Cambridge, UK, Sept. 15–18 2008.

[16] Simon A. Winder and Matthew Brown. Learning local image descriptors. In *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.