Learning Convolutional Filters for Interest Point Detection

Andrew Richardson Edwin Olson

***Abstract*— We present a method for learning efficient feature detectors based on in-situ evaluation as an alternative to hand-engineered feature detection methods. We demonstrate our in-situ learning approach by developing a feature detector optimized for stereo visual odometry.**

Our feature detector parameterization is that of a convolutional filter. We show that feature detectors competitive with the best hand-designed alternatives can be learned by random sampling in the space of convolutional filters and we provide a way to bias the search toward regions of the search space that produce effective results. Further, we describe our approach for obtaining the ground-truth data needed by our learning system in real, everyday environments.

I. Introduction

In this paper, we present a method to automatically learn feature detectors that perform as well as the best hand-designed alternatives and are tailored to the desired application. Most current feature detectors are hand-designed. Creating feature detectors by hand allows the designer to leverage human intuition to create intricate and efficient detectors that would be hard to replicate automatically. However, many feature detectors tend to be reused across multiple applications with subtle differences that could be leveraged to improve performance. Automatically-learned detectors have the potential to exploit these differences and improve application performance, as well as help us better understand the properties of top-performing feature detectors. Such a learning process, however, comes with its own challenges, such as defining a sufficiently general parameterization in which good detectors can be found and tractably exploring such a parameter space.

This work approaches automatic feature detector learning by learning fast and effective feature detectors based on *convolutional filters*. Convolutional filters make up an expressive space of feature detectors yet possess favorable computational properties. We specifically learn these detectors for use in stereo Visual Odometry (VO), an application in which the full set of feature detector invariances, such as affine invariance, are not required to achieve good motion estimates and efficiency is key. Unlike feature detection and matching evaluations limited to planar scenes, these detectors are learned on realistic video sequences in everyday environments to ensure the relevance of the learned results. Learning high-performance feature detectors also requires a good source of training data, which can be difficult or expensive to obtain. We detail our method for generating ground truth data — instrumenting an environment with 2D fiducial markers known as AprilTags [17]. These markers can be robustly detected with low false-positive rates, allowing us to extract features with known data association across an entire video. We can then solve for the global poses of all cameras and AprilTags and use this information to evaluate the motion estimate when using natural feature detections. Importantly, we also use this global reconstruction to reject any natural feature detections on or near AprilTags, ensuring that we do not bias the learning process to detect fiducial markers.

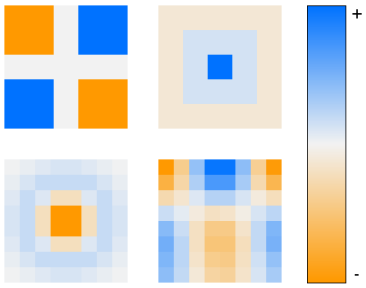


Fig. 1: When using convolutional feature detectors, which filter is best? Reasonable examples include the corner and point detectors used in [8] (top, both) and a Difference-of-Gaussians (DOG) filter (bottom left) like those used for scale-space search in SIFT [12]. We automatically generate filters for feature detection to improve the 3D motion estimation of a stereo visual odometry system. Bottom right: the most accurate filter in this work.

The feature detectors learned with our method are efficient and accurate. Processing times are comparable to FAST, a well-known method for feature detection, and reprojection errors for stereo visual odometry are lower than those with FAST in most cases [18]. In addition, because our filters use a simple convolutional structure, processing times are reduced by both increases in CPU clock rates and SIMD vector instruction widths.

The main contributions of this paper are:

1) We propose a framework for learning a feature detector designed to maximize performance of a specific application.
2) We propose convolutional filters as a family of feature detectors with good computational properties for general-purpose vector instruction hardware (e.g. SSE, NEON).
3) We present a sampling-based search algorithm that can incorporate empirical evidence that suggests where high

The authors are with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48104, USA {chardson,ebolson}@umich.edu http://april.eecs.umich.edu

quality detectors can be found and tolerate both a large search space and a noisy objective function. We detail both the search algorithm and a set of experiments used to steer the search towards effective portions of the search space.

4) We present a method to evaluate the performance of learned filters with easily-collected ground truth data. This enables evaluation of the end application in arbitrary 3D environments.

In Section II, we discuss background material and prior work. We then discuss the general principles of our method for detector learning in Section III. Section IV focuses on the specifics of our application, stereo visual odometry, and ground-truthing method. Finally, our experiments and evaluation are presented in Section V.

II. Background

Many feature detectors have been designed to enhance feature matching repeatability and accuracy through properties such as rotation, scale, lighting, and affine-warp invariance. Some well known examples include SIFT [12], SURF [2], Harris [9], and FAST [18]. While SIFT and SURF aim to solve the scale-invariant feature detection problem, Harris and FAST detect single-scale point features with rotation invariance at high framerates. Other detectors aim to also achieve affine-warp invariance to better cope with the effects of viewpoint changes [14].

Many comparative evaluations of feature detectors and descriptors are present in the literature [14], [16]. Breaking from previous research, Gauglitz et al. evaluated detectors and descriptors specifically for monocular visual tracking tasks on video streams [7]. This evaluation is beneficial, as continuous motion between sequential frames can limit the range of distortions, such as changes in rotation, scale, or lighting, that the feature detectors and descriptors must handle, especially in comparison to image-based search methods that can make no such assumptions. Our performance-analysis mechanism is similar; however, whereas [7] focused on rotation, scale, and lighting changes for visual tracking, we focus on non-planar 3D scenes.

An alternative to hand-crafted feature detectors and descriptors is automated improvement through machine learning. FAST, which enforces a brightness constraint on a segment of a Bresenham circle via a decision tree, is a hand-designed detector that has been optimized for efficiency via ID3 [18]. An extension of FAST, FAST-ER, used simulated annealing to maximize detection repeatability [19]. In contrast to these approaches, we focus on learning a detector to improve the output of our target application using a method with a low and nearly-constant feature detection time.

Detector-learning work by Trujillo and Olague used genetic programming to assemble feature detectors composed of primitive operations, such as Gaussian blurring and equalization [20]. Their results are promising, though the training dataset size was small. Additionally, they attempt to maximize detector repeatability, whereas our method is focused on the end-to-end system performance of our target application.

In addition to the detector-learning methods, descriptor learning methods like those from Brown et al. learn local image descriptors to improve matching performance [4]. They use discriminative classification and a ground-truthed 3D dataset. Their resulting descriptors perform significantly better than SIFT on the ROC curve, even with shorter descriptors. As this paper focuses on *detector* learning, we use a common *descriptor* for all detectors. This descriptor uses a standard pixel-patch representation and allows an even comparison between all detectors evaluated in this work.

III. Learning a Feature Detector

Feature detector learning requires three main components: a parameterization for the detector, an evaluation metric, and a learning algorithm. The parameterization defines a continuum of detectors, ideally capable of describing the range from fixed-size point or corner detectors to scale-invariant blob detectors, as well as concepts like "zero mean" filters. The evaluation harness computes the error of a proposed detector, which we want to minimize. Given these components, we can construct a method to generate feature detectors that maximize our learning objective. While iterative optimization through gradient or coordinate descent are popular ways to solve such problems, these approaches are problematic in learning feature detectors due to the high-dimensional search space and noise in the objective function. Random sampling allows us to evaluate far more filters than with iterative methods, find good filters despite numerous local minima, and develop an intuition for the constraints on the filter design that yield the best performance.

A. Detector parameterization

We parameterize our feature detector as a convolutional filter [15]. This is an attractive representation due to the convolutional filter's flexibility and the ability to leverage signal processing theory to interpret or constrain the qualities of a detector. In addition to a convolutional filter's flexibility, these filters can also be implemented very efficiently on vector instruction hardware, such as Intel SSE, AVX, and ARM NEON. This hardware is commonly available in modern smartphone processors, as rich media applications can benefit significantly from SIMD parallelism.

We want to find the convolutional filter that yields the most accurate result for our target application. This is different from the standard metrics for feature detector evaluation like repeatability, as the best features may not in fact be detectable under all rotations. Our objective function (which we minimize) measures the error in the motion estimate against ground truth. This is in contrast to methods which maximize an approximation of end-to-end performance like repeatability [19], [20]. The advantage of our approach is the potential to exploit properties specific to the application. In stereo visual odometry, for example, edges that are vertical from the perspective of the camera are easy to match between

(a) 4×4 DCT basis set (b) 4×4 Haar basis set

Fig. 2: Basis sets representing each frequency-domain transform for (a) the Discrete Cosine Transform (DCT) and (b) the Haar Wavelet Transform on a 4×4 image patch. Each patch corresponds to a single coefficient in frequency space. The top left basis corresponds to the DC component of the filter. Note that we use 8×8 filters in this work.

the left and right frames due to the epipolar geometry constraints. This property is not captured by standard measures like repeatability, so methods which use these measures cannot be expected to exploit them.

Our detection method can be summarized as follows:

1) Convolve with the filter to compute the image response
2) Detect points exceeding the filter response threshold, which is updated at runtime to detect a constant user-specified number of features
3) Apply non-extrema suppression using the filter response over a 3×3 window

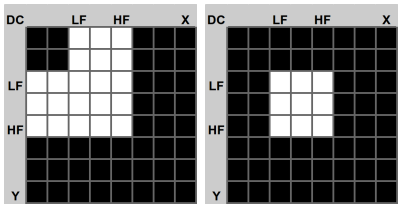
This work focuses on 8×8 convolutional filters. A naïve parameterization would simply specify the value of each entry in the filter, a space of size $\mathbb{R}^{64}$. In the following section, we detail alternative parameterizations which allow us to learn filters which both perform better and require less time to learn.

B. Frequency domain parameterizations

Within the general class of convolutional filters, we parameterize our feature detectors with frequency domain representations. In this way, we can apply meaningful constraints on the qualities of these filters that would not be easily specified in the spatial domain. In doing so, we learn about the important characteristics for successful feature detectors built from convolutional filters.

We use the Discrete Cosine Transform (DCT) and Haar Wavelet transform to describe our filters [1], [15]. Unlike the Fast Fourier Transform (FFT), the DCT and Haar Wavelets only use real-valued coefficients and are known to represent image data more compactly than the FFT [3]. This compactness is often exploited in image compression and allows us to sample candidate filters more efficiently. These transformations can be easily represented by orthonormal matrices and computed through linear matrix products.

In this work, we make use of three representations for filters — a straightforward pixel representation, the Discrete Cosine Transform (DCT), and the Haar Wavelet Transform.



(a) Bandpass (b) Block-diagonal

Fig. 3: Example frequency-domain filter constraints for 8×8 filters. Coefficients rendered in black are suppressed (forced to zero). White coefficients can take any value. A typical bandpass filter is shown in (a). We propose the use of the block-diagonal region of support in (b), which exhibited superior performance in our tests.

Figure 2 shows a set of basis patches for the two frequency-domain representations. The pixel values of a filter in the spatial domain can be uniquely described by a weighted linear combination of these basis patches, where the weights are the *coefficients* determined by each frequency transformation. For convenience, we refer to both the spatial values of the filters (pixel values) and frequency coefficients as $\mathbf{w}$ for the remainder of this paper.

C. Error minimization

In our application, our goal is to find a convolutional filter which yields the best motion estimate for a stereo VO system. We represent the error function that evaluates the accuracy of a proposed filter by $E(\mathbf{w})$. As described further in Section IV-B, our error function is the mean reprojection error of the ground truth data, the four corners of the AprilTags, using the known camera calibrations and the camera motion estimated using the detector under evaluation.

While iterative optimization via gradient or coordinate descent methods is a straightforward approach for learning with an error function, we found that such iterative methods get caught in local minima too frequently. We propose instead to learn detectors by randomly-sampling frequency coefficient values while varying the size and position of a coefficient mask that zeroes all coefficient values outside of the mask. We refer to our mask of choice as a block-diagonal constraint, as illustrated in Figure 3. We also evaluate the performance of sampling with bandpass constraints and sampling raw pixel values via the naïve approach. In all cases, sampled values are taken from a uniformly-random distribution in the range [-127,127].

The constraints illustrated in Figure 3 are defined by low and high frequency cutoffs, which define the filter's bandwidth. The filters shown have low and high frequency cutoffs of 0.250 and 0.625, respectively[1]. Thus, the filters have a bandwidth of 0.375.

Iteratively-optimizing filters can be very expensive. Steepest descent methods that use the local gradient of the error function require at least n error evaluations for square filters

[1]Note that we use frequency ranges normalized to the interval (0, 1)

of width $\sqrt{n}$. After gradient calculation, multiple step lengths may be tried in a line-search minimization algorithm. At a minimum, $n+1$ error evaluations are required for every update. Coordinate descent methods require at least $2n$ evaluations to update every coefficient once. Both methods require more calculations in practice. Because the error surface contains a high number of local minima, step sizes must be small and optimization converges quickly. This results in a great deal of computation for only small changes to the filter. We found that 95% of our best randomly-sampled filters did not reduce their error significantly after iterative optimization. Many did not improve at all.

In contrast to iterative methods, computing the error for a new filter only requires one evaluation. The result is that in the time it would take to perform one round of gradient descent for an 8×8 filter, we can evaluate a minimum of 65 random samples.

IV. Stereo Visual Odometry

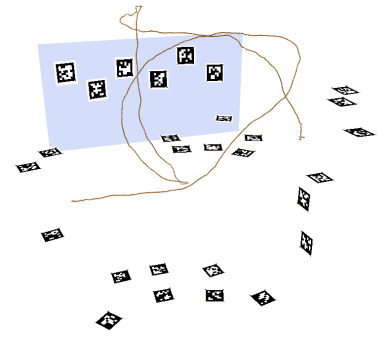
In this section, we describe aspects of our application domain, stereo visual odometry. In principle, our in-situ training methods apply to other stereo visual odometry pipelines and other applications. We use a custom stereo visual odometry pipeline for feature detector learning. Prior work in this area has yielded accurate and reliable results with high-framerate VO using corner detectors [13], [11]. A number of system architectures are possible and have been demonstrated in the literature, including monocular [11], [5] and stereo approaches [13], [8].

A. Visual Odometry overview

Our approach to stereo visual odometry can be divided into a number of sequential steps:

1) **Image acquisition** - 30 Hz hardware-triggered frames are paired using embedded frame counters before stereo rectification via bilinear interpolation.
2) **Feature detection** - Features are detected in grayscale images with non-extrema suppression. Zero-mean, 9×9 pixel patch descriptors are used for all features.
3) **Matching** - Features are matched between paired images using a zero-mean Sum of Absolute Differences (SAD) error score. To increase robustness to noise, we search over a +/-1 pixel offset when matching. Unique matches are triangulated and added to the map. Previously-mapped features are projected using their last known 3D position and matched locally.
4) **Outlier rejection** - We provide robustness to bad matches with both Random Sample Consensus (RANSAC) and robustified cost functions during optimization (specifically, the Cauchy cost function with $b = 1.0$) [6], [10].
5) **Motion estimation** - We initialize the motion estimate to the best pose from RANSAC. We then use nonlinear optimization to improve the point positions and camera motion estimates, iterating until convergence.

The result is an updated 3D feature set and an estimate of the camera motion between the two sequential updates.



(a) Reconstructed ground truth trajectory

(b) Reprojected ground truth features

Fig. 4: Ground truth reconstruction using AprilTags. Stereo camera trajectory (orange) in (a) is reconstructed using interest points set on the tag corners determined by the tag detection algorithm. Shaded region (blue) corresponds to the scene viewed in (b). The mask overlays (red) in (b) are the result of reprojecting the tag corners using the ground truth reconstruction and are used to ensure that no features are detected on the AprilTags added to the scene. Example feature detections from the best filter learned in this work are shown for reference (green).

B. Ground truth using AprilTags

We compute our ground truth camera motion by instrumenting the scene with AprilTags and solving a global nonlinear optimization over all of the tags and camera positions. Specifically, we treat the four tag corners as point features with unique IDs for global data association. During learning, we explicitly reject any feature detections on top of or within a small window around any AprilTag so that we do not bias the detector learning process. In other words, our system rejects detections that would otherwise occur because of the AprilTags in the scene so that we learn to make use of the *natural* features in the environment. This is illustrated in Figure 4b, where red overlays correspond to regions where all feature detections are discarded. By using the reprojected AprilTag positions, we can reject features on AprilTags even when a tag is not detectable in the current frame.

Once the ground truth has been computed, we can compute the error of a motion estimate computed with an arbitrary feature detector. For every sequential pair of poses in the dataset, we compute the 3D position of the AprilTag corners with respect to the first of the two poses, transform from the first to the second pose's reference frame with the motion estimate from the arbitrary feature detector, and compute the reprojection error of these points in the images from the

(a) Office (b) Conference room

(c) Cubicle (d) Lobby

Fig. 5: Images from the four datasets used in this work. AprilTag masks described previously are shown in red.

second pose. $E(\mathbf{w})$ is the mean reprojection error over all pairs of poses in the dataset using this method. It measures how well, on average, the ground truth features are aligned with their observations when using the candidate detector.

V. Experiments

Our experiments focus on randomly-sampling filters under different constraints and computing the mean reprojection error when using these filters across multiple datasets. In addition, we compare both error and computation time to existing and widely-used feature detectors.

A. Implementation Details

Training and testing take place on four datasets between 23 and 56 seconds in length with 30 FPS video. These datasets were collected in various indoor environments, including an office, conference room, cubicle, and lobby. In each case, we randomly selected 15 seconds of video for training. Figures 4 and 5 show camera trajectories and imagery from these datasets. Our stereo rig uses two Point Grey FireFly MV USB2.0 color cameras at a resolution of 376×240. Experiments were run on a pair of 12-core 2.5 GHz Intel Xeon servers, each with 32 GB of memory. Sampling 5,000 filters takes approximately 7 hours at 9.7 seconds per filter.

In contrast to the substantial computational resources used in learning, our target application is limited to the compute available on a typical mobile robot. A mobile-grade processor such as the OMAP4460, a dual-core ARM Cortex-A9 device, used as an image preprocessing board, is a compelling and scalable computing solution to our needs. With a convolution implementation optimized via vector instructions, specifically ARM NEON, we can detect around 300 features per 376×240 image in 3.65 ms for a 8×8 filter. In comparison, the ID3-optimized version of the FAST feature detector performs similarly, requiring 3.20 ms. For both methods, we dynamically adjust the detection threshold to ensure the desired number of features are detected even as the environment changes.

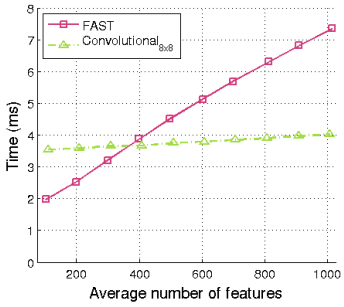
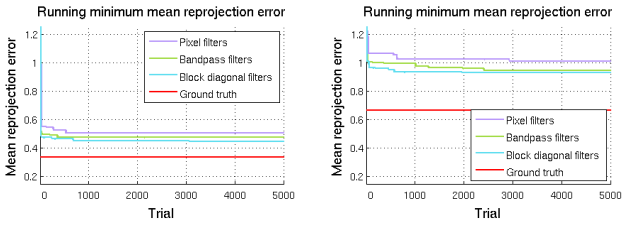


Fig. 6: Time comparison between FAST-9 and an 8×8 convolutional filter feature detector. Times represent the combined detection and non-extrema suppression time and were computed on the PandaBoard ES (OMAP 4460) over 30 seconds of video with 376x240, grayscale imagery.



(a) Desk dataset (b) Conference room dataset

Fig. 7: Mean reprojection error for the best filter sampled so far (running minimum error) over 5,000 samples. Ground truth system error shown in red.

While both methods are efficient enough for real-time use, the difference in computation time for a large number of features is dramatic. Figure 6 shows the runtime as a function of the desired number of features for both methods. This time measurement includes detection and non-extrema suppression. While both methods show a linear growth in computation time, the growth for the convolutional methods is much slower than for FAST. This is because the convolution time does not change as the detection threshold is reduced. The linear growth is due only to non-extrema suppression. This result is especially important for methods where the desired number of detections is high [11], [8].

B. Randomly-sampled filters

We evaluated our approach by running a suite of random-sampling experiments for block-diagonal filters with both the DCT and Haar parameterization. We also compare to sampled bandpass and pixel filters. Figure 7 shows the error of the best filter sampled so far as 5,000 filters are sampled. In all four datasets, the pixel and bandpass filters never outperformed the best block-diagonal filter. The final filters of each type and from each dataset[2] are shown in Figure 8.

Figure 9 shows the error distributions for each of the three constraints on the conference room dataset. For each experiment, 5,000 filters were randomly generated with the

[2]Final coefficients available at umich.edu/~chardson/icra2013feature.html

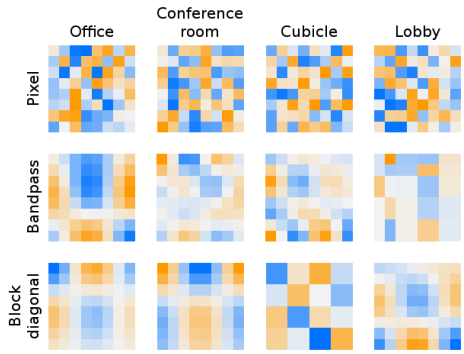
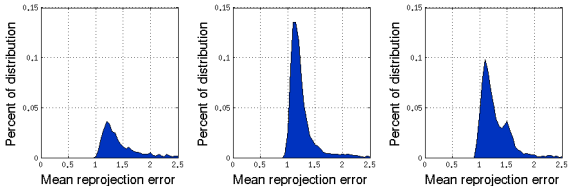
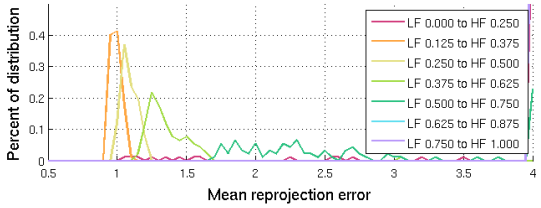


Fig. 8: Best learned filter for every type and dataset combination. Best viewed in color.

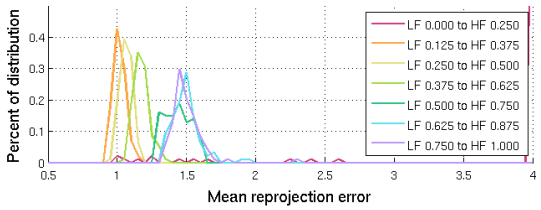


(a) Pixels (b) Bandpass (c) Block-diagonal

Fig. 9: Histogram of mean reprojection errors for three filter generation methods on the conference room dataset. While randomly-generated bandpass filters yield better performance, on average, than filters with uniformly-random pixel values, block-diagonal filters have both better average performance and a lower error for the best filters. 79% and 72% of sampled bandpass and block-diagonal filters, respectively, had errors under 2.5 pixels, while only 30% of random pixel filters had such low errors.



(a) Distributions for sampled DCT block-diagonal filters



(b) Distributions for sampled Haar block-diagonal filters

Fig. 10: Distributions for block-diagonal filters with a bandwidth of 0.250 as a function of the low frequency cutoff on the conference room dataset. For both (a) and (b), the filters which include the DC and vertical/horizontal edge components (LF cutoff of 0) have reprojection errors greater than 4 pixels 84% and 87% of the time for DCT and Haar filters, respectively. Beyond the DC components, only the DCT filters with low-frequency cutoff of 0.5 or greater have such large reprojection errors. The remaining distributions progress smoothly from low error (left) to high error (right) as the frequency increases. Best viewed in color.

appropriate set of constraints. For the bandpass and block-diagonal constraints, we generated filters for all combinations of the DCT or Haar transform, low frequency cutoff and filter bandwidth, defined previously. Of the 27 combinations of low and high-frequency mask cutoffs available for both DCT and Haar filters of size 8×8 (in total, $\binom{8}{2} - 1$ combinations each for DCT and Haar), only 6 of them include the DC component and, in the case of the block-diagonal filters, the vertical and horizontal edge components.

From these plots, it is clear that limiting the search for a good filter through the bandpass and block-diagonal sampling constraints significantly improved the percentage of filters which yield low reprojection errors. Our interpretation of this result is that filters are very sensitive to nonzero values for specific frequency components. By strictly removing these components in 21 of the 27 constraint combinations, the average filter performance improves significantly.

Figure 10 shows separate distributions for block-diagonal filters for each of the possible low-frequency cutoffs for filters with the most narrow filter bandwidth (0.250). One plot is shown for each frequency transformation (DCT and Haar). From these figures, it is clear that filters perform significantly worse when the DC and edge coefficient values are not zero. Beyond that, we see a trend of low error for low-frequency filters, and an increasing error as frequency increases. Finally, the DCT filters seriously degrade when they begin to contain high frequency components; however, the Haar filters do not. Our interpretation of this result is that, because Haar basis patches are not periodic, a simple step transition in an image will often result in a unique maxima. In contrast, the periodic DCT basis patches will yield multiple local maxima, causing a cluster of detections around edges. Similar trends exist for filters with higher bandwidths.

The performance of the best sampled block-diagonal filters is compared to baseline methods such as FAST, Shi-Tomasi, a Difference of Gaussians filter, and filters used by Geiger et al in Table I. The best result from every testing dataset (row) is shown in bold. All detector evaluations were performed with the same system parameters: detect 300 features after non-extrema suppression and filtering with the AprilTag masks, use RANSAC and a Cauchy robust cost function ($b = 1.0$), etc. These parameters were set via parameter sweeps using the FAST feature detector on the office and conference room dataset. As such, these represent best-case conditions for FAST. Mean values over 25 trials are reported due to variability induced by RANSAC. The differences in the means between the trained filters and FAST were statistically significant in 10 of the 12 cases with p values

Testing dataset	Filters trained on specified dataset				Linear baseline methods			Nonlinear baseline methods		
	Office	**Conf rm.**	**Cubicle**	**Lobby**	**DOG**	**Geiger Corner**	**Geiger Blob**	**FAST**	**Shi-Tomasi**	**SURF**
Office	0.447	**0.466**	0.471	0.468	40.281^{+}	0.492	0.595	0.470	2.060	1.322*
Conf rm.	0.981	0.929	0.996	0.981	1.505	1.047	1.218	**0.953**	1.141	1.584*
Cubicle	1.292	**1.142**	1.134	1.368	2.962	2.131	4.042	1.441	4.550	0.795*
Lobby	1.593	**1.552**	1.628	1.482	1.974	1.573	1.927	1.654	1.938	2.032*

TABLE I: Testing error on each dataset using the learned feature detectors and baseline methods. Reported numbers are mean values over 25 trials to compensate for the variability of RANSAC, except for the training errors (gray). Bolded values are the best result for every row. *SURF generated features adjacent to AprilTags that could not be easily filtered out because of SURF's scale invariance. As such, the SURF results are not considered a fair comparison to the other methods. $^{+}$Large errors are the result of data association failures with the specified features.

less than 0.01 for a two-tailed t-test.

These results reinforce the notion that learned convolutional filters can compete with nonlinear detection methods, like the FAST feature detector. Only on the conference room dataset did FAST perform better than a learned filter. On average, learned filters had a lower reprojection error than FAST by a small amount, 0.05 pixels. For other baseline methods, such as Shi-Tomasi, the improvement in reprojection error was substantial. Note that while SURF outperformed all methods on the cubicle datasets, this is due to SURF detections adjacent to AprilTags that cannot be rejected as easily due to SURF's scale invariance.

For the linear baseline methods, the results vary greatly. Geiger et al's corner filter performs the best of the three, and yet it and the other linear baselines perform quite poorly on the cubicle dataset, unlike the learned filters. Interestingly, these linear detectors (or an equivalent 8×8 filter) are simply a few of the convolutional filters that could have been learned in our framework.

These results also suggest that dataset choice, not learned filter, was the best predictor of testing errors. The cubicle dataset had a high error in a number of cases. From inspecting the video stream, this is not surprising — the cubicle is generally featureless except for a few smooth edges and a narrow strip at the top where the camera sees over the cubicle wall.

VI. Summary

We have presented a framework for automatically learning feature detectors that can be efficiently computed on modern architectures and result in performance that is generally better than existing methods, sometimes substantially so. By sweeping over frequency-domain constraints on the filters during sampling, we learn detectors that outperform obvious alternatives and prior work. In addition, our results indicate that the best detectors are typically those that respond primarily to the lowest non-DC frequency components. These learned detectors perform well on all datasets, despite only using one dataset during training.

Acknowledgements

This work was supported by U.S. DoD Grant FA2386-11-1-4024.

References

[1] N. Ahmed, T. Natarajan, and K. Rao. Discrete cosine transform. *Computers, IEEE Transactions on*, 100(1):90–93, 1974.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

[3] T. Bose, F. Meyer, and M. Chen. *Digital signal and image processing*. J. Wiley, 2004.

[4] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE transactions on pattern analysis and machine intelligence*, pages 43–57, 2010.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1052–1067, 2007.

[6] M. Fischler and R. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis andd automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981.

[7] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International Journal of Computer Vision*, pages 1–26, 2011.

[8] A. Geiger, J. Ziegler, and C. Stiller. StereoScan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium*, Baden-Baden, Germany, June 2011.

[9] C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[11] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

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

[13] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, pages 1–17, 2010.

[14] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1):43–72, 2005.

[15] T. Moon and W. Stirling. *Mathematical methods and algorithms for signal processing*, volume 204. Prentice hall, 2000.

[16] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3D objects. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 800 – 807 Vol. 1, 2005.

[17] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[18] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[19] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, November 2008.

[20] L. Trujillo and G. Olague. Automated design of image operators that detect interest points. *Evolutionary Computation*, 16(4):483–507, 2008.