Parameter Learning for Improving Binary Descriptor Matching

Bharath Sankaran, Srikumar Ramalingam, and Yuichi Taguchi

Abstract—Binary descriptors allow fast detection and matching algorithms in computer vision problems. Though binary descriptors can be computed at almost two orders of magnitude faster than traditional gradient based descriptors, they suffer from poor matching accuracy in challenging conditions. In this paper we propose three improvements for binary descriptors in their computation and matching that enhance their performance in comparison to traditional binary and non-binary descriptors without compromising their speed. This is achieved by learning some weights and threshold parameters that allow customized matching under some variations such as lighting and viewpoint. Our suggested improvements can be easily applied to any binary descriptor. We demonstrate our approach on the ORB (Oriented FAST and Rotated BRIEF) descriptor and compare its performance with the traditional ORB and SIFT descriptors on a wide variety of datasets. In all instances, our enhancements outperform standard ORB and are comparable to SIFT.

I. INTRODUCTION

Feature matching is a key component in several vision tasks such as object detection, object recognition, and structure-from-motion. State-of-the-art approaches to these problems rely on robustly matching descriptors that are costly to compute and match. This led to the advent of binary descriptors that are fast to compute and match [1] [2] [3]. These are particularly crucial in mapping and localization tasks for autonomous navigation, where computational speed is critical. The computational speed and efficiency of binary descriptors are attributed to the following properties:

- Binary descriptors are computed by pairwise pixel intensity comparisons in a given image patch. Pixel comparisons are faster to compute than gradient operations, which are used in gradient based descriptors such as SIFT [4] and SURF [5].
- Binary descriptors are matched using Hamming distance metrics that are faster to compute than the L2 metric used for gradient based descriptors.

ORB [1] is one of the binary descriptors that is two orders of magnitude faster than SIFT [4], without losing much on the performance with respect to keypoint matching. We propose several extensions to improve the performance of ORB descriptor by learning a few parameters, without making the descriptor computation any slower. Our approach is readily extensible to any binary descriptor, since all binary descriptors only vary by the way the pairwise pixels are sampled from a given image patch. For instance BRIEF [6] descriptor uses random pairs. BRISK [2] uses a hand-crafted sampling pattern, whereas ORB [1] and FREAK [7] use pairs that are learned from data. In order to explain our approach effectively we provide a brief introduction to the computation of the ORB descriptor below.

A. The Basic ORB Computation and Matching

Let us consider the problem of matching two keypoints $k_1$ and $k_2$. Consider a small patch of dimension $p \times p$ centered at these keypoints. ORB considers 256 pairs of pixels $(p_i, q_i), i = \{1, \ldots, 256\}$ in the patch and performs simple binary tests. An example of pairs selected for such binary tests are shown in Figure 1. These 256 different pairs are chosen based on a greedy algorithm that looks for highly informative pairs learned from PASCAL dataset [8]. Let $I(p)$ denote the intensity value of the pixel $p$. The binary test performed on the intensity values is

$$b_i = \begin{cases} 1 & \text{if } I(p_i) > I(q_i), \\ 0 & \text{otherwise}. \end{cases}$$  

The entire $256 \times 1$ feature vector $f_1 \in \mathbb{B}^{256}$ for the patch at keypoint $k_1$ is given by $f_1 = [b_1, \ldots, b_{256}]^T$. Two keypoints $k_1$ and $k_2$ are matched by looking at the Hamming distance $\mathcal{H}(f_1, f_2)$ between the feature vectors constructed at these keypoints:

$$\mathcal{H} = \sum_{i=1}^{256} |f_1(i) - f_2(i)|$$

II. RELATED WORK

Fast similarity search has garnered significant attention in the recent years to enable real-time applications in various kinds of data like image, video, and audio. In image matching in particular, the advent of a wide variety of binary descriptors [6], [1], [2], [7] has led to substantial gains in matching speeds for real-time applications without a huge compromise on performance. Though binary descriptors perform similar to traditional descriptors in easy matching cases, they are
sensitive to challenging conditions such as lighting, viewpoint, and scale variations.

In order to improve the matching accuracy of binary descriptors there have been earlier approaches that have proposed the idea of learning weights. The general idea tries to improve matching accuracy by learning weights such that the Hamming distance of correct matches is lower in comparison to wrong matches. Fan et al. [9] demonstrate a lookup table approach to compute fast weighted Hamming distances that demonstrate equivalent matching speeds in comparison to the standard Hamming distance. Similarly, weighted Hamming approaches have been used for feature ranking. For instance, Zhang et al. [10] introduce a dynamic bit level weighting method for ranking binary codes to reduce the number of instances that receive the same Hamming distance. This approach to binary code ranking, though more discriminative than a regular weighted Hamming matcher, is not computationally efficient. In our approach to weighted matching of binary descriptors we avoid compromising on matching speed by performing a forward and reverse consistency check, i.e., a source to target and target to source match. We later prune these matches to get the final set of accurate matches.

Comparing single pixels in binary descriptors causes the representation to be sensitive to noise and minor image distortions. Patch based approaches have been proposed in order to be robust. For instance LATCH [11] is a novel binary descriptor that focuses on comparing mini-patches in order to increase the spatial support of binary tests. To construct the descriptor they use triplets of mini-patches instead of pairs. The set of triplet of patches are learned from data around a given keypoint. The LATCH descriptor uses a predefined set of 512 triplets where similarity between patches are measured with sum of squared differences (SSD). In contrast, in our approach we let the threshold parameter handle the variation in noise and image distortion that the binary descriptor is susceptible to.

Most methods developed for improving descriptor matching accuracy have primarily focused on learning better similarity metrics. Apart from weighted matching of descriptors, there are also approaches that focus on improving the expressiveness of the descriptor. Zagoruyko and Komodakis [12] train Convolutional Neural Networks (CNN) for matching image patches. Their network is a 2 channel CNN with the two top layers consisting of single channel fully connected layers. This architecture was tested in a siamese, pseudo-siamese, and non-siamese framework. This approach to image matching was shown to outperform conventional descriptor based image matching approaches. Though these approaches have higher matching accuracy, they have lower computational and matching speeds in comparison to binary feature matching approaches.

III. PROBLEM FORMULATION

A. Proposed Extensions

Based on the basic binary descriptor computation and matching framework, we propose three extensions to the descriptor computation and performance. We demonstrate these extensions over the standard ORB descriptor, while using the same pairs as the standard ORB descriptor. In the first extension we propose to use a weighted Hamming distance matcher instead of the one that considers uniform weights for all the 256 different binary tests. We learn a weight vector \( w = [w_1, \ldots, w_{256}]^T \) and match two keypoints using weighted Hamming distance:

\[
H_w = \sum_{i=1}^{256} w_i |f_1(i) - f_2(i)|
\]

These weights can be learned as shown in Section III-B.

The binary tests are usually performed after smoothing the images. Despite this, the binary tests are sensitive to lighting and viewpoint variations. We propose to use a threshold \( T \in \mathbb{R} \) in the binary tests as follows:

\[
b_i = \begin{cases} 1 & \text{if } I(p_i) - I(q_i) > T, \\ 0 & \text{otherwise.} \end{cases}
\]

Learning this threshold is more involved than learning weights. We explain this second extension in Section III-C. Then as the third extension, we propose to learn both the weights and threshold, which is explained in Section III-D.

B. Learning The Weights

Given some training data \( \mathcal{D} = \{x_i, y_i\}, i = \{1, \ldots, n\} \), we would like to learn some weights so that the weighted Hamming distance for correct matches is smaller than the distance for the incorrect matches. Here, \( x_i \) and \( y_i \) are 256×1 binary vectors for \( n \) correct keypoint matches. We formulate the problem of learning the weights using the standard max-margin network learning [13] in the following manner:

\[
\min_{w, b, \epsilon} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \epsilon_i \\
\text{s.t.} \\
H_w(x_i, y_i) + b \leq -1 + \epsilon_i \\
H_w(x_i, y_j) + b \geq 1 - \epsilon_i, \forall j \neq i \\
\epsilon_i \geq 0
\]

Here \( \epsilon_i \) is the slack variable and \( C \) is the soft margin parameter in standard max-margin network learning algorithms. \( w \) is the set of weights that we learn and \( b \) is the bias term. To learn the weights we use two negative cases of \( H_w(x_i, y_j) \) for every positive case of \( H_w(x_i, y_i) \). The positive case of \( H_w(x_i, y_i) \) is the correct match between source and target descriptor, \( x_i \) and \( y_i \). This is given from the training data. The two negative cases used for learning are the target descriptors which have the smallest and second smallest Hamming distance to the source descriptor, where \( j \neq i \).

C. Learning The Threshold

The optimization problem for threshold learning can be formulated as follows. Given some training data \( \mathcal{D} = \{d_{i1}, d_{i2}\}, i = \{1, \ldots, n\} \), we would like to learn a threshold \( T \in \mathbb{R} \). Here, \( d_{i1} \) and \( d_{i2} \) refer to 256×2 matrices storing the intensity values for 256 pairs of pixels used for building the binary descriptors at two different matching keypoints. We formulate the learning problem as shown below:
\[
\begin{align*}
\min_{T, b, \epsilon} & \sum_{i=1}^{n} \epsilon_i \\
\text{s.t.} & \quad H(x_i, y_i) + b \leq -1 + \epsilon_i \\
& \quad H(x_i, y_j) + b \geq 1 - \epsilon_i, \forall j \neq i \\
x_i(k) = & \quad \arg \min_{x_i(k) \in \{0, 1\}} x_i(k)(d_{i1}(k, 1) - d_{i1}(k, 2) - T) \\
y_i(k) = & \quad \arg \min_{y_i(k) \in \{0, 1\}} y_i(k)(d_{i2}(k, 1) - d_{i2}(k, 2) - T) \\
T & \geq -255.0 \\
T & \leq 255.0
\end{align*}
\]

The threshold \( T \) takes only integer values, because the error does not change for any intermediate real values. We can thus perform a brute-force search for different threshold values. Also, we only learn a single threshold value for the entire 256 bit vector.

D. Combined Weight and Threshold Learning

To combine both the weight and threshold learning we formulate the optimization as follows. Given some training data \( D = \{d_{i1}, d_{i2}\}, i = \{1, ..., n\} \), we would like to learn the weight vector \( w \in \mathbb{R}^{256} \) and the threshold \( T \in \mathbb{R} \). We formulate the learning problem as shown below:

\[
\begin{align*}
\min_{w, b, T, \epsilon} & \quad \frac{1}{2} w^T w + C \sum_{i=1}^{n} \epsilon_i \\
\text{s.t.} & \quad H_w(x_i, y_i) + b \leq -1 + \epsilon_i \\
& \quad H_w(x_i, y_j) + b \geq 1 - \epsilon_i, \forall j \neq i \\
x_i(k) = & \quad \arg \min_{x_i(k) \in \{0, 1\}} x_i(k)(d_{i1}(k, 1) - d_{i1}(k, 2) - T) \\
y_i(k) = & \quad \arg \min_{y_i(k) \in \{0, 1\}} y_i(k)(d_{i2}(k, 1) - d_{i2}(k, 2) - T) \\
T & \geq -255.0 \\
T & \leq 255.0
\end{align*}
\]

The above problem is non-convex and it is difficult to get an optimal solution. We can fix the threshold \( T \) to different integer values and this makes the optimization problem convex, similar to the weight learning method explained in Section III-B. The constraint involving \( \arg \min \) leads to the non-convexity. Exploiting the integer nature of the threshold values, it can be learned via a brute-force search.

IV. Experiments and Observations

We evaluated the descriptor matching accuracy of our method and other approaches on four different datasets: the Oxford affine covariant regions dataset\(^1\) first introduced in [14], the AMOS (Archive of Many Outdoor Scenes)\(^2\) [15], the KITTI stereo dataset\(^3\) [16], and the CA VE dataset\(^4\) [17]. We evaluated our approach by comparing the RANSAC refined inlier ratio with other descriptors in the Oxford, AMOS and CA VE datasets. For Oxford and AMOS datasets we also evaluate our approach with models trained on other datasets like Cornell Multiview dataset\(^5\) [18] and the KITTI stereo dataset. For CA VE we only evaluate SIFT and ORB against a model trained on the KITTI stereo dataset. Finally for KITTI, we only compare our approach against the standard ORB.

A. The Datasets

1) Oxford dataset: The Oxford dataset has images with scale, viewpoint and lighting variations. A few examples from this dataset are shown in Figure 2. In the Oxford dataset, the ground truth homographies between a single source image and multiple target images are provided along with the dataset. We use this information to find all matching keypoint pairs in source and target images across the dataset. From these keypoint pairs, we extract the descriptor differences and use this information to train our max-margin model to learn the weights while optimizing thresholds as discussed in Section III-D. During the testing phase, we

\[^2\]http://amos.ca.wustl.edu/dataset
\[^3\]http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php
\[^4\]http://www.cs.columbia.edu/CAVE/databases/multispectral/
\[^5\]http://www.cs.cornell.edu/projects/p2f/
independently detect 500 keypoints in source and target images (without using the ground truth homography). Hence in our training/testing phase we use the same images but during the training phase the keypoints are transformed from source to target using the ground truth homographies. There is no special reason for chosing 500 keypoints in our experiments, except that feature detectors like ORB are very efficient in extracting 500 keypoints from VGA images in many real-time applications. We match these keypoints using our algorithm and report the RANSAC refined inlier ratio. We compare our results with other descriptors like SIFT and vanilla ORB. We also train models on other datasets like KITTI and Cornell and test them on the Oxford dataset.

2) The AMOS dataset: The AMOS dataset is a publicly available archive of outdoor scenes taken from fixed cameras over multiple days. From this dataset we assembled an illumination variant dataset with images taken from the same camera position over multiple times of the day over multiple days of the month. An example of such a sequence of a single day is shown in Figure 3. We use models trained on the Oxford lighting data subset, the KITTI dataset and the Cornell Multiview dataset to evaluate our approach on the AMOS dataset. Hence our testing set comes from a different distribution compared to our training sets.

Similar to the Oxford evaluation, for AMOS we independently detect 500 keypoints in the source and target image and match them using SIFT, ORB and our approach. We compute the RANSAC refined inlier ratio for comparison with other approaches like SIFT and ORB.

3) The KITTI dataset: The KITTI dataset is an autonomous driving platform dataset [16] that was introduced as a standard benchmark for computer vision problems like stereo, optical flow, visual odometry, 3D object detection and 3D tracking. A sample stereo pair from the dataset is shown in Figure 4. The KITTI dataset provides a standard train and test set. Ground truth disparity for all stereo pairs are provided with the dataset. In the training phase we extract the descriptors from matching keypoint pairs in stereo images using the disparity information provided with the dataset. For evaluation we compare the RANSAC refined inlier ratio for stereo matches computed using ORB and our approach.

4) CAVE dataset: The CAVE dataset contains multispectral images that are used to emulate a Generalized Assorted Pixel (GAP) camera. The images are a wide variety of real-world materials and objects. Examples are shown in Figure 5. We only evaluate some subsets of the CAVE dataset where ORB keypoints could be detected in both source and target images. For the CAVE dataset we separate the dataset into equally sized but mutually independent training and testing sets. Here the training and testing sets are extracted from the same class of images but do not share the same images.

5) The Cornell multiview dataset: The Cornell dataset is a multiview city scale dataset that contains images taken from different viewpoints, different locations and different cameras. The dataset was first introduced in [18]. The training data comes with ground truth bundler [19] data that can be used to get the pixel location of 3D points seen by multiple cameras. We use this information to learn a model on points extracted from multiple image/pixel pair combinations across the dataset. In our experiments we specifically train a model on the Dubrovnik dataset. We use the model trained on this dataset to evaluate matching on the Oxford and AMOS dataset.

B. Matching results for Oxford dataset

In this evaluation we detect the same number of keypoints in SIFT, the threshold and the non-threshold version of the ORB detector. The inlier percentage is the mean inlier percentage across all target images from the Oxford dataset. Each subset of the Oxford dataset has 5 target images. \( \mathcal{H}_w(x_i, y_j) \) is ORB matched with the weighted Hamming distance without any threshold optimizations. \( \mathcal{H}_w(x_i, y_j) \) is the threshold optimized ORB, matched with a vanilla Hamming distance matcher. \( \mathcal{H}_{wτ}(x_i, y_j) \) is the model that was trained on the threshold optimized ORB with the weighted Hamming distance matcher. \( \mathcal{H}_{wτ}(x_i, y_j) \) trained on data from the Oxford dataset, the Cornell dataset and the KITTI dataset are denoted as Oxford \( \mathcal{H}_{wτ}(x_i, y_j) \), Cornell \( \mathcal{H}_{wτ}(x_i, y_j) \) and KITTI \( \mathcal{H}_{wτ}(x_i, y_j) \) respectively. The best results are \textbf{boldfaced} and the second best results are \textbf{boldfaced} in blue color. We show the accuracy in Table I. The weighted Hamming distance provides better accuracy compared to naive Hamming distance. In general, the weighted Hamming distance along with the threshold provides the best accuracy. The Cornell and KITTI datasets are larger than the Oxford dataset. The weights and thresholds learned using these datasets also provide good accuracy on the Oxford dataset.

From our experiments we can see that our learned model outperforms the vanilla ORB descriptor in all datasets and
Fig. 3. **AMOS dataset.** The images show the progression of an entire day from a single viewpoint. The leftmost image in the first row is the first image taken at the start of the day. The rightmost image in the second row is the last image taken at night.

<table>
<thead>
<tr>
<th>SIFT</th>
<th>Graffiti</th>
<th>Boat</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66.2%</td>
<td>58.52%</td>
<td>71.56%</td>
</tr>
<tr>
<td>ORB</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB ($H_w(x_i, y_j))$</td>
<td>61.14%</td>
<td>52.93%</td>
<td>76.91%</td>
</tr>
<tr>
<td>ORB (Oxford $H_w(x_i, y_j))$</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB (Cornell $H_w(x_i, y_j))$</td>
<td>59.55%</td>
<td>53.64%</td>
<td>82.89%</td>
</tr>
<tr>
<td>ORB (KITTI $H_w(x_i, y_j))$</td>
<td>63.31%</td>
<td>54.24%</td>
<td>77.57%</td>
</tr>
<tr>
<td></td>
<td>60.77%</td>
<td>55.10%</td>
<td>82.04%</td>
</tr>
</tbody>
</table>

Table I: **OXFORD DATASET EVALUATION RESULTS**

<table>
<thead>
<tr>
<th>SIFT</th>
<th>Graffiti</th>
<th>Boat</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66.2%</td>
<td>58.52%</td>
<td>71.56%</td>
</tr>
<tr>
<td>ORB</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB ($H_w(x_i, y_j))$</td>
<td>61.14%</td>
<td>52.93%</td>
<td>76.91%</td>
</tr>
<tr>
<td>ORB (Oxford $H_w(x_i, y_j))$</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB (Cornell $H_w(x_i, y_j))$</td>
<td>59.55%</td>
<td>53.64%</td>
<td>82.89%</td>
</tr>
<tr>
<td>ORB (KITTI $H_w(x_i, y_j))$</td>
<td>63.31%</td>
<td>54.24%</td>
<td>77.57%</td>
</tr>
<tr>
<td></td>
<td>60.77%</td>
<td>55.10%</td>
<td>82.04%</td>
</tr>
</tbody>
</table>

Table II: **AMOS DATASET EVALUATION RESULTS**

<table>
<thead>
<tr>
<th>SIFT</th>
<th>Graffiti</th>
<th>Boat</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>45.01%</td>
<td>58.52%</td>
<td>71.56%</td>
</tr>
<tr>
<td>ORB</td>
<td>45.03%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB ($H_w(x_i, y_j))$</td>
<td>45.86%</td>
<td>52.93%</td>
<td>76.91%</td>
</tr>
<tr>
<td>ORB (Oxford $H_w(x_i, y_j))$</td>
<td>46.29%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td></td>
<td>45.86%</td>
<td>52.93%</td>
<td>76.91%</td>
</tr>
</tbody>
</table>

Table III: **KITTI STEREO DATASET EVALUATION RESULTS**

<table>
<thead>
<tr>
<th>SIFT</th>
<th>Graffiti</th>
<th>Boat</th>
<th>Light</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>66.2%</td>
<td>58.52%</td>
<td>71.56%</td>
</tr>
<tr>
<td>ORB</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB ($H_w(x_i, y_j))$</td>
<td>61.14%</td>
<td>52.93%</td>
<td>76.91%</td>
</tr>
<tr>
<td>ORB (Oxford $H_w(x_i, y_j))$</td>
<td>55.43%</td>
<td>47.28%</td>
<td>66.83%</td>
</tr>
<tr>
<td>ORB (Cornell $H_w(x_i, y_j))$</td>
<td>59.55%</td>
<td>53.64%</td>
<td>82.89%</td>
</tr>
<tr>
<td>ORB (KITTI $H_w(x_i, y_j))$</td>
<td>63.31%</td>
<td>54.24%</td>
<td>77.57%</td>
</tr>
<tr>
<td></td>
<td>60.77%</td>
<td>55.10%</td>
<td>82.04%</td>
</tr>
</tbody>
</table>

**D. Matching results for KITTI Stereo dataset**

For the KITTI evaluation we independently detect 1500 keypoints on source and target images and match them. We evaluate our approach against the vanilla ORB descriptor and matcher. We perform the evaluation on the test set provided along with the dataset. The result of the model evaluation for descriptor matching is shown in Table III. The inlier percentage is the mean inlier percentage across all 200 stereo pairs from the KITTI test set. We observe that by learning weights and thresholds, we achieve an increase in the accuracy of the matching. In our evaluation we also noticed that our threshold optimized approach does comparably to ORB on the simple cases and better on the harder cases.

**E. Matching results for CAVE Multispectral dataset**

Similar to the Oxford evaluation, we independently detect 500 keypoints in the source and target images and match them. We evaluate our approach, i.e., learning the threshold optimized weighted Hamming matcher $H_w(x_i, y_j)$ against SIFT and ORB. We use three independent models trained on the Oxford dataset, Cornell dataset and KITTI dataset respectively. These are the same models used in the Oxford evaluation. The inlier percentage is the mean inlier percentage across all 20 target images from the AMOS dataset. We show the results of our approach in Table II. As we can once again see, our approach outperforms both SIFT and ORB. By learning weights and thresholds from large datasets such as Cornell and KITTI, we observe an improvement in the accuracy of the matching algorithm.

**F. Discussion**

Through our experiments we made the following observations about the characteristics of our threshold optimized weighted Hamming matcher.

- Our matcher had fewer false positives as compared to the traditional ORB descriptor and matcher. This led to...
a lower number of overall matches but a higher number of accurate matches. Hence the RANSAC refined inlier ratio was higher.

- Our results show that the weights can be easily learned on a large publicly available dataset to allow better generalization.

**TABLE IV**

**CAVE DATASET EVALUATION RESULTS**

<table>
<thead>
<tr>
<th>CST</th>
<th>CL</th>
<th>OP</th>
<th>JB</th>
<th>WC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>94.51%</td>
<td>90.71%</td>
<td>72.34%</td>
<td>84.73</td>
</tr>
<tr>
<td>ORB</td>
<td>96.99%</td>
<td>80.45%</td>
<td>71.83</td>
<td>75.06%</td>
</tr>
<tr>
<td>(H_{w_t}(x_i, y_j))</td>
<td>99.32%</td>
<td>81.47%</td>
<td>81.08%</td>
<td>75.96%</td>
</tr>
</tbody>
</table>

**G. Implementation Details**

We implemented our algorithms using OpenCV. The weights for the weighted Hamming distance matcher were learned with a linear SVM using the LIBSVM library [20].

We exploit the integer nature of the threshold values by using brute-force search to solve for the threshold values. The threshold adjusted descriptors are computed using a speed optimized implementation similar to the OpenCV ORB implementation.

**V. CONCLUSION AND FUTURE WORK**

We have demonstrated an approach to learn weights and thresholds to improve descriptor matching performance for binary descriptors. We demonstrated our approach on the ORB descriptor, but our method is readily applicable to other binary descriptors. For the threshold optimization, we present a search algorithm by exploiting the fact that we only need to consider the integer threshold values. In the future, we plan to develop an automatic algorithm to optimize and solve the non-convex threshold learning problem. We also intend to learn a 256 bit threshold vector to capture finer variation while matching. The threshold based Hamming distance without the weights runs at the same speed as traditional ORB, since there are no added computations other than bit comparisons. The weighted Hamming distance (under our current non-optimized implementation) is at least 15x slower than the regular Hamming distance. However, there are approaches to make it as fast as regular Hamming distance using a lookup table [9]. Given that our current implementation is in OpenCV, we intend to make our code available.

**Acknowledgments:** We thank Jay Thornton, Arvind Raghunathan, Gim Hee Lee, Alan Sullivan, Khalid Yousif, anonymous reviewers, and the area chair for useful feedback and support.

**REFERENCES**


