Learning Visual Feature Descriptors for Dynamic Lighting Conditions

Nicholas Carlevaris-Bianco and Ryan M. Eustice

***Abstract*—In many robotic applications, especially long-term outdoor deployments, the success or failure of feature-based image registration is largely determined by changes in lighting. This paper reports on a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features. We demonstrate that, by tracking feature points in time-lapse videos, one can easily generate training data that captures how the visual appearance of interest points changes with lighting over time. This training data is used to learn feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where Euclidean distance provides good discrimination between matching and non-matching image patches. Results showing that the learned descriptors increase the ability to register images under varying lighting conditions are presented for a challenging indoor-outdoor dataset spanning 27 mapping sessions over a period of 15 months, containing a wide variety of lighting changes.**

I. Introduction

Standard hand-designed visual features such as scale invariant feature transform (SIFT) [1] and speeded up robust features (SURF) [2] detect key-points in an image and then describe the local visual appearance of these key-points as a vector. Image registration can then be performed by matching the key-points between images by comparing the $\mathcal{L}_2$ distance between the descriptors. In order for matching to be successful, the key-point detector and descriptors must be at least partially robust to common image variations such as scale, rotation, view-point, and lighting changes. Invariance with respect to scale and rotation are usually accounted for at the feature detection stage, where key-points will be detected at a canonical scale and orientation. The description stage then focuses on representing the appearance of the local region around the key-point such that the descriptor is discriminative while being robust to view-point and illumination changes.

In this paper we focus on increasing the illumination robustness of feature point description to lighting changes. Hand-designed descriptors such as SIFT and SURF have limited lighting invariance—often allowing for affine transformations in image intensity by considering the gradient of intensity, and through other mechanisms such as mean subtraction and normalization. However, in general, the change in appearance caused by lighting affects the image intensity in a complex, nonlinear way.

Fig. 1: Sample images from 3 of the 500 locations in the North Campus dataset used for testing. Imagery was collected in 27 sessions over the course of 15 months with lighting conditions ranging from early morning to just after dusk. The success or failure of feature-based image registration in this dataset is largely determined by the similarity of lighting conditions.

In many robotic applications, the success or failure of feature-based image registration is largely determined by changes in lighting. This is especially true for medium to long-term outdoor applications, where the scene structure has not changed dramatically, but images separated by even a few hours may be unmatchable due to cyclical changes in lighting. This phenomenon is illustrated in Fig. 1, which shows example imagery from three different locations in our experimental dataset. In this dataset, only a small fraction of the possible matches are successfully registered using standard features, largely because of cyclical changes in lighting.

In this work, we seek to learn a feature descriptor that is more robust to changes in local image appearance caused by lighting (Fig. 2). To observe how the local appearance of image patches changes under dynamic lighting conditions, we first track key-points and their associated image patches through time-lapse video using a representative training dataset. We then train a feature descriptor using matching and non-matching pairs of image patches sampled from these patch tracks. A contrastive cost function is used so that matching patches are mapped close together (in terms of Euclidean distance in feature space) while separating non-matching patches. The resulting descriptor is more robust to the types of lighting variation observed in the training data.

*This work was supported in part by the National Science Foundation under award IIS-0746455, the Office of Naval Research under award N00014-12-1-0092, and Ford Motor Company via the Ford-UM Alliance under award N015392.

N. Carlevaris-Bianco is with the Department of Electrical Engineering & Computer Science, University of Michigan, Ann Arbor, MI 48109, USA carlevar@umich.edu.

R. Eustice is with the Department of Naval Architecture & Marine Engineering, University of Michigan, Ann Arbor, MI 48109, USA eustice@umich.edu.

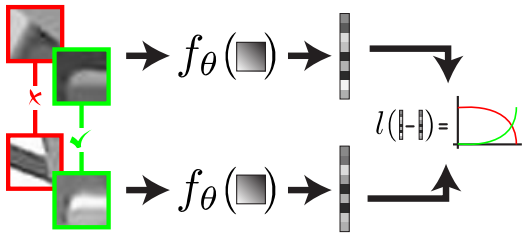


Fig. 2: Illustration of the learning method. Pairs of image patches labeled either as matching (green) or non-matching (red) are supplied as input to a feature descriptor function, $f_{\theta}(\cdot)$, parameterized by $\boldsymbol{\theta}$, that maps the input patch to a feature vector. A contrastive cost function, $l(\cdot)$, based on the Euclidean distance between the feature vectors, encourages matching feature vectors to be close together in feature space while encouraging non-matching features to be far apart. By learning parameters $\boldsymbol{\theta}$ that minimize this cost function, we produce a mapping to a feature space where Euclidean distance captures the similarity and differences amongst the training pairs. By training with data that includes variation due to changes in lighting, the feature descriptor learns to be robust to lighting variation.

The remainder of this paper is outlined as follows: In Section II, we discuss existing work related to the proposed method. The descriptor learning method is described in Section III. Section IV contains details of the training process, including the collection of training data. Experimental results are provided in Section V. Finally a discussion and concluding remarks are provided in Sections VI and VII.

II. Related Work

Given the limitations of existing visual feature descriptors, several proposed methods address the difficulties in matching images collected under varying lighting conditions at a systems level. In a mapping and navigation context, both Konolige and Bowman [3] and Churchill and Newman [4] add new example views or visual "experiences" when the current view cannot be registered against previous views. This addresses the problem of changing lighting by capturing several examples of how a location might look under different lighting conditions. Similarly, in Johns and Yang [5, 6], locations are modeled with a collection of features observed at different points in time. These works are mostly orthogonal to the proposed method, and would benefit from features that are more robust to lighting change, because better features reduce the number of samples needed to model a location.

Several recent works have investigated whole image place recognition under changing appearance conditions, including [6–9]. In Lategahn et al. [8], a set of standard descriptor "building blocks" is defined. Place recognition performance is then optimized by searching the space of possible descriptors constructed from these building blocks. Neubert et al. [7] attempt to predict how a location will look at a different point in time by learning a mapping between appearance codewords. They then perform place recognition between the current image and the predicted image. The formulation, however, focus on changes between two distinct states (e.g., summer and winter) and not continual changes such as those caused by lighting. In Milford et al. [9], whole image place recognition is performed over extreme changes in lighting from day to night by aggressively down-sampling and contrast normalizing the images before comparison.

In this work, we focus on geometric registration through point correspondence as opposed to whole image place recognition. For some applications, like loop-closure detection in metric mapping, even if one can recognize places under a high degree of lighting variation, it may not be useful if one cannot extract a metric estimate of the motion between the camera views [10]. It is worth noting that the feature descriptors learned using our proposed method could be used in a bag-of-words model [11, 12] for place recognition, however, evaluating if this would improve the robustness with respect to lighting remains future work.

Many methods have been proposed that leverage machine learning to improve the performance of feature descriptors [10, 13–18]. In Babenko et al. [13], feature matching is cast as a binary classification problem where one attempts to determine if two image patches do or do not match. Boosting is then used with a set of simple hand-designed features to learn a classifier appropriate for a specific domain. In Hua et al. [14], Winder and Brown [15], Winder et al. [16], and Brown et al. [17], the parameters of fixed descriptor pipelines (often a variant of the DAISY descriptor [19]) are optimized to improve descriptor performance. Similarly, in Stavens and Thrun [20], the parameters of standard descriptors, including SIFT, are optimized for specific domains. Ranganathan et al. [10] use the fine vocabulary method of [21] to learn a probability distribution over visual words in an attempt to capture which visual words can be produced by the same scene feature under various lighting conditions. Standard place recognition and feature matching are then reformulated to account for the learned distribution. Both Philbin et al. [22] and Shakhnarovich [23] learn an embedding on top of SIFT features. This is similar to the proposed method except that we learn an embedding directly from the raw pixel input as opposed to on top of a hand-designed feature descriptor. The recent work by Trzcinski et al. [18], which uses boosting to learn a binary descriptor, is most similar to our proposed method in that it learns a descriptor directly from raw pixel data in a supervised setting. However, our proposed method differs in its learning method, descriptor model, and also in its focus on robustness to changes in lighting.

To learn an illumination robust feature descriptor we employ a training scheme referred to as a "Siamese" network [24–28], with the goal of minimizing a contrastive cost function [25, 26, 28] that encourages a nonlinear mapping to a lower-dimensional space where matching features are close together and non-matching features are far apart in Euclidean distance. This goal is often referred to as embedding learning, manifold learning, or distance metric learning.

Siamese networks have been employed in a wide range of applications including signature verification in Bromley et al. [24], face recognition in Chopra et al. [25], and object recognition in Hadsell et al. [26] and Mobahi et al. [27]. An especially compelling result was presented in Taylor et al.

[28] where a system was trained that could recognize similar human poses while being highly invariant to other distractors, including changes in lighting.

Beyond Siamese networks, auto-encoder frameworks can also be used to learn nonlinear embeddings as shown in Hinton and Salakhutdinov [29] and [30]. However, with auto-encoders the goal is to produce a lower-dimensional embedding that can be decoded with minimal reconstruction error, which does not necessarily produce embeddings where Euclidean distance is useful for discrimination [28]. Salakhutdinov and Hinton [31] provide an interesting model that blends a Siamese network with an auto-encoder for regularization.

The contrastive cost function employed here is just one option for learning an embedding. In Goldberger et al. [32], a linear model is optimized in order to minimize a probabilistic version of k-nearest neighbors classification error. A probabilistic loss function based on Kullback-Leibler divergence (KLD) is provided in Hinton and Roweis [33].

III. Method

In this section, we first describe the Siamese network framework that can be used to learn a wide variety of feed-forward descriptor models (Fig. 2). We then discuss the specific feature descriptor models considered in this work. As in [13–17], we focus on the description of the image patch associated with key-points provided by an existing key-point detector[1]. We assume that the detector provides us with a pixel location, a scale, and optionally a canonical orientation to allow for rotation invariance[2]. Given this information we extract an appropriate patch from the image for each key-point. The image patch then becomes the input for the learned descriptor. At this point we assume that we have pairs of patches labeled as either matching or non-matching. We detail how these pairs can be easily generated in §IV.

A. Learning A Feature Descriptor

First, we define a feature descriptor that maps an image patch $\mathbf{x}$ to a feature vector $\mathbf{y}$ as

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x}),$$

where $\boldsymbol{\theta}$ parameterizes the descriptor. During training we work with pairs of training examples that are known to be matching or non-matching. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two training image patches. The current function is then used to describe each patch,

$$\mathbf{y}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i) \text{ and } \mathbf{y}_j = f_{\boldsymbol{\theta}}(\mathbf{x}_j).$$

We then consider the squared Euclidean distance in feature space

$$d_{ij}^2 = \|\mathbf{y}_i - \mathbf{y}_j\|_2^2.$$

Using the contrastive cost function from [26],

$$l_{\boldsymbol{\theta}}(\mathbf{y}_i, \mathbf{y}_j) = \begin{cases} s_{ij} d_{ij}^2, & \text{if matching} \\ \max\left(1.0 - d_{ij}^2,\, 0\right), & \text{if non-matching} \end{cases} \quad (1)$$

[1]We use the SURF detector throughout our experiments.

[2]In our experiments we do not exploit the canonical orientation as we focus on robotic applications where the imagery does not undergo large rotations.

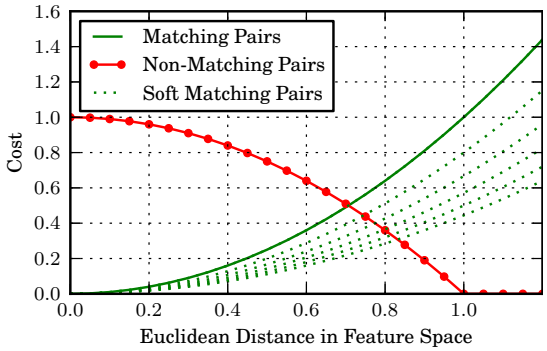


Fig. 3: Plot of the contrastive cost function in (1). The cost for matching pairs is shown in green and the cost for non-matching pairs in red. The dashed green lines show the matching cost function using similarity weighting (2) with $\alpha = 1/8$ h for $|t_i - t_j| = [2, 4, 5, 6, 10]$ h.

but with the similarity score s_{ij} between the matching pairs set as proposed in [28][3] (since we are training with temporal sequences), we define s_{ij} based on the difference in time between when the two patches were observed,

$$s_{ij} = \frac{1}{1 + \alpha |t_i - t_j|}, \quad (2)$$

where t_i and t_j are the observation times of the training patches in hours and α is a scale factor controlling the time scale of the similarity weight. In our experiments, we selected $\alpha = 1/8$ h. Taylor et al. [28] experimentally demonstrated that this "soft" similarity allows the embedding to better capture the temporal similarity in appearance. They also experimentally showed that this soft similarity improved training results. The resulting cost function (1) is illustrated in Fig. 3.

If we consider a training set of N training pairs the learning objective becomes

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} l_{\boldsymbol{\theta}}\left(\mathbf{y}_i^n, \mathbf{y}_j^n\right). \quad (3)$$

This objective can be optimized using stochastic gradient descent, the details of which are described in §IV.

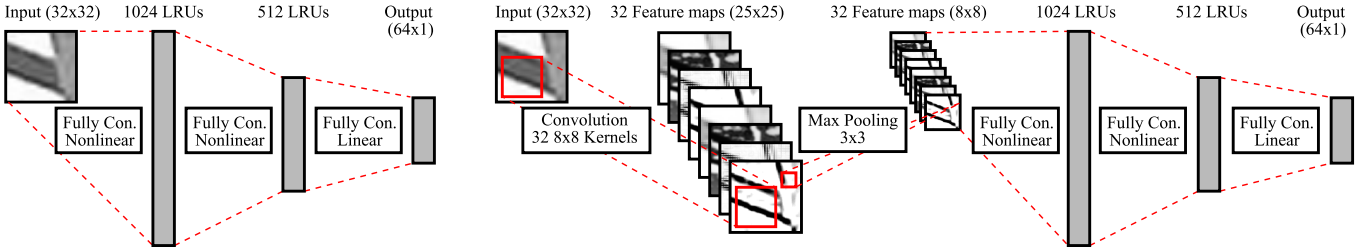
B. Feature Descriptor Models

In our experiments we consider two standard model classes for the learned feature descriptor; a multi-layer perceptron (MLP) and a convolutional multi-layer perceptron (CMLP) [34] (Fig. 4). The MLP consists of multiple fully-connected layers, each performing a nonlinear transformation on the output of the previous layer. If we denote the input to each hidden layer as $\mathbf{h}_{i-1}$ and the output as $\mathbf{h}_i$ then

$$\mathbf{h}_i = g\left(\mathrm{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i\right),$$

where W_i is a matrix defining a linear transform, $\mathbf{b}_i$ is a bias vector, and $g(\cdot)$ is a nonlinear activation function applied in an elementwise fashion to its input vector. This layer is parameterized by $\boldsymbol{\theta}_i = [\mathrm{W}_i, \mathbf{b}_i]$, which it contributes to the

[3]Hadsell et al. [26] set $s_{ij} = 1$ to treat all matching pairs evenly.



(a) Multi-Layer Perceptron (MLP)

(b) Convolutional Multi-Layer Perceptron (CMLP)

Fig. 4: Feature descriptor models.

parameters of the overall model. For the first layer the input will simply be the raw image patch as a vector $\mathbf{h}_0 = \mathbf{x}$.

The CMLP expands upon the MLP by adding convolutional and pooling layers. The convolutional layers exploit the fact that the statistics of natural images can be considered stationary over the location in the image. Instead of learning the parameters of a function of the whole image, weights are learned for kernels that are convolved with the image to produce a number of feature maps. This greatly reduces the number of parameters in the model without significantly reducing its representational capacity. For a detailed description we refer the reader to [34]. The pooling layers perform a spatial subsampling that reduces the size of the input for subsequent layers and provides invariance to small translational shifts in the input. In the proposed models we use non-overlapping max pooling, which performed slightly better than mean pooling. It is interesting to note that the CMLP structure is very similar to that of many hand-designed feature descriptors [1, 2, 19], which often include a convolution filtering stage (e.g., computing orientated gradients) and a pooling stage (e.g., spatial binning or averaging).

In both the MLP and CMLP we use rectifying nonlinearity, referred to as a linear rectified unit (LRU) [35],

$$s(x) = \max(x, 0),$$

which we found to be quicker to train than hyperbolic tangent, or sigmoid nonlinearities. Additionally, both the MLP and CMLP have a linear output layer.

IV. Training

In this section, we describe how we produce training data by tracking interest points in time-lapse videos. We also provide the details of the stochastic gradient descent learning.

A. Generating Training Data

Many methods have been proposed to generate a training set of image patches. In [14–17], 3D reconstructions are used to establish correspondence between patches in the source images. In [22], image-to-image feature-based matching with outlier rejection is used in order to generate training data, since they seek only to learn an encoding on top of existing feature descriptors. Images with known pose are used in [8]. It is also possible to generate sequences of image patches by tracking interest points in video [20, 36] or by sliding a window through static images [37].

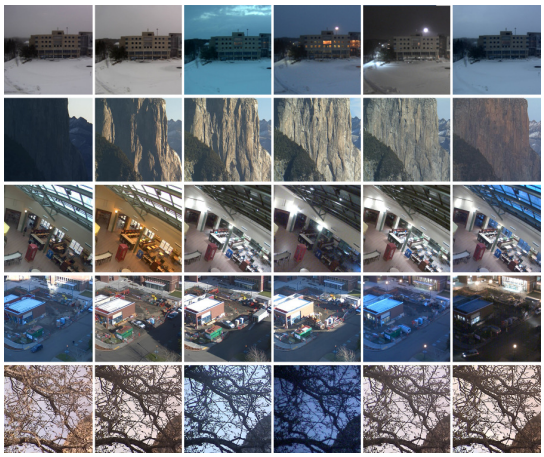


Fig. 5: Sample images from five locations in the webcam dataset. Note that the images have been sub-sampled so the difference in time between images is larger than the 20 minutes used during patch extraction

In this work we elect to generate patches by tracking interest points in video. In order to capture the changes in appearance caused as the lighting changes with time, we use time-lapse videos. To generate these videos we downloaded imagery from stationary webcams at a fixed rate. In total, 230 different webcam locations were used, including mostly outdoor natural and urban scenes, as well some indoor scenes. Imagery was downloaded every 20 minutes for 72 hours. Sample imagery from five locations is shown in Fig. 5. Of the 230 locations, 184 were used to generate training data, 23 for validation and 23 for testing in §V.

1) Tracking interest points in time-lapse videos: Given a sequence of webcam frames, we track features through time as follows:

- We detect interest points in each incoming image. In our experiments we use the SURF detector, however, any detector that provides location and scale would be acceptable. (Note that we do not use the SURF descriptor for tracking). One could also use the canonical orientation of an interest point detector in order to achieve some degree of rotation invariance; however, we do not in our experiments as our target application uses a ground robotic platform that does not undergo large rotations.

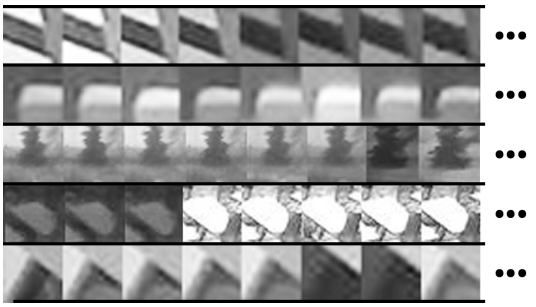


Fig. 6: Sample patch tracks extracted from the webcam dataset. Note that the tracks will be of varying lengths, and only a subsections are shown here.

- For each interest point we attempt to associate it with an existing track. Because the imagery is collected from a static viewpoint we can use several simple criteria, similar to those proposed in [16, 17]. First, the interest point must be within 5 pixels of the most recent observation of the track. Second, the scale of the interest point must be within $\pm 50\%$ of the most recent observation's scale. Third, the difference in time between a new patch and the most recent observation of a track can be no more than 1 hour. These criteria are often enough to uniquely associate a new patch with an existing track. If there there are still multiple candidates, we select the track that minimizes
$$\frac{r + r_t - \|\mathbf{x} - \mathbf{x}_t\|_2}{2\max(r, r_t)},$$
where r and r_t are the radii of the interest point and track, respectively, and $\mathbf{x}$ and $\mathbf{x}_t$ are the locations of the interest point and track, respectively. If we cannot find a valid existing track, a new track is created based upon that patch.
- After processing each image we consider the current tracks. Tracks that have been updated recently are kept for association in future images. Tracks that have not been updated in an hour are no longer updated and are wrapped up and saved.
- Because we wish to emphasize the temporal change in the dataset, we subsample the final tracks by two, increasing the time between each sample in the track from 20 minutes to 40 minutes.

The results of this process are illustrated in Fig. 6. Using this processing pipeline on the entire webcam dataset produced approximately 3.1 million feature tracks (2.5 million for training, 0.3 million for validation and 0.3 million for testing) with an average of approximately 5 patches per track.

2) Generating training pairs from tracks: Given a set of patch tracks, it is easy to generate a very large set of matching and non-matching pairs for training. Starting with all feature tracks we randomly sample pairs of tracks without replacement. From a pair of tracks we then randomly select two matching pairs (one from each set) and two non-matching pairs (from between the two sets). This produces an even number of matching and non-matching pairs in the dataset. We repeat this process until all patch tracks have been used at least once. This ensures that each track is used.

Given the combinatorially huge number of possible pairs of tracks, and possible pairs within each track, this process can be repeated multiple times. During training we continuously sample new pairs.

3) Augmenting the training data with viewpoint variation: The webcam dataset does a good job of capturing the changes patches undergo with respect to lighting; however, because the videos are captured from static locations, they do not contain any view-point variance. To account for the lack of viewpoint variance we augment the patches extracted from the webcam dataset using the viewpoint variant patches provided in Brown et al. [38]. This dataset provides an additional 0.9 million patch pairs (which we divide evenly between training, testing, and validation). Another option would be to use the existing image patches with synthetic affine warps, which has been shown to produce good results in [39].

B. Training Descriptor Models

Batch stochastic gradient descent was employed in order to optimize the model parameters in (3). We used a batch size of 1000 pairs with a learning rate of $\lambda = 0.1$ and momentum of $\beta = 0.9$, producing an update procedure at step k of

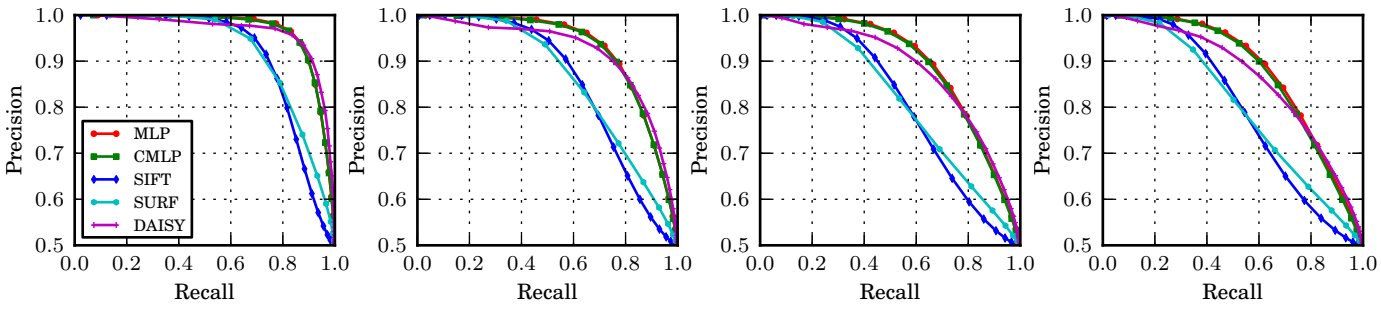
$$\mathbf{v}_{k+1} = \beta \mathbf{v}_k - \lambda \frac{\partial \mathcal{L}_k}{\partial \boldsymbol{\theta}_k}$$
$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \mathbf{v}_{k+1},$$

where $\frac{\partial \mathcal{L}_k}{\partial \boldsymbol{\theta}_k}$ is the gradient of the objective function (3) with respect to the parameters $\boldsymbol{\theta}$ over the kth batch of training data. Training was implemented using Theano [40], which allows for automatic differentiation of the objective function and GPU-based evaluation of the feature descriptor models.

V. Experimental Results

We evaluate the proposed feature descriptor on two datasets. The first consists of 23 webcam locations not used during training. This dataset is used to evaluate how temporal changes in lighting affects matching. The second dataset consists of data collected by a ground robot and allows us to compare the descriptor's performance in a challenging real-world environment.

In addition to the proposed descriptors, we compare against SIFT [1], SURF [2], and DAISY [19]. For the DAISY descriptor we use learned parameters provided by [16], specifically the "T1-4-2r8s" version as it has an output dimension of 68 and computation time comparable with our learned descriptors. In comparison, SIFT has a dimension of 128 while SURF and both of the learned descriptors produce 64 dimensional vectors. We also attempted to compare with another learned descriptor, DIRD [8]; however, DIRD was optimized with respect to whole image place recognition and was not effective for point-to-point geometric image registration. In order to focus on the properties of the descriptors, the same key-points (which were detected using the SURF detector) were used for all feature descriptors.



(a) 1 h Between Images (b) 4 h Between Images (c) 8 h Between Images (d) 12 h Between Images

Fig. 7: Precision and recall curves for pairs from the 23 test webcam locations. By comparing the precision-recall curves for pairs approximately 1 h, 4 h, 8 h, and 12 h apart, we see that the performance of the proposed learned features degrades gracefully as the time between images increases, especially in the region above 90% precision.

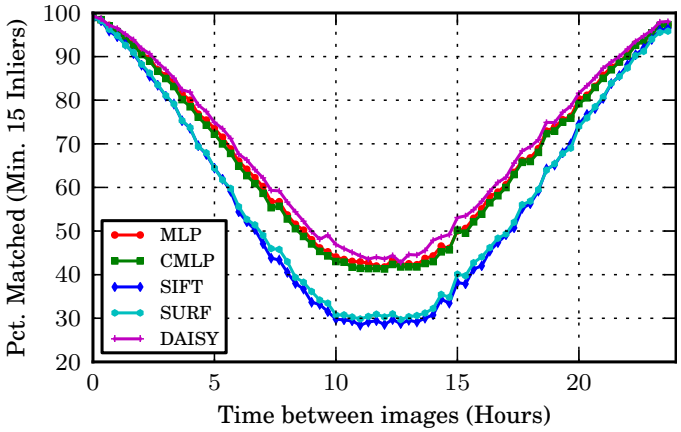


Fig. 8: Matching results for webcam dataset. Results are averaged from exhaustive pairwise matching at 23 webcam locations.

A. Webcam Dataset

We first explore the performance of the feature descriptors using imagery from 23 webcam locations that were not used during training. Because the webcam data was collected frequently, it allows us to evaluate performance with respect to the time between images.

Using matching and non-matching pairs from this test set we sweep out the precision and recall curve for a descriptor by classifying points as matching or non-matching with a varying distance threshold. We see in Fig. 7 that when the time between image pairs is small, all of the methods perform well, with the learned descriptors and DAISY having the best performance. However, as we increase the time between image pairs, the two learned methods degrade gracefully, maintaining good precision-recall curves in the challenging region around 12 hours. Note that this is especially true in the region above 90% precision where we would like to operate so that matching is not overwhelmed by outliers.

To evaluate the features in an image registration context we attempt to register all pairs of images at each location that were collected within 24 hours of each other. The ability to match images in this situation is driven primarily by the change in lighting throughout the day. So over the course of 24 h, the ability to register images will start high, and gradually reduce as the time between images increases, hitting a minimum at about 12 h before increasing as time-of-day lighting conditions return to those most similar after 24 h. For indoor locations the patterns might be less distinct, but still, many indoor locations go through similar cycles caused by working hours and light through windows.

When matching features we perform nearest neighbor matching based on Euclidean distance and only include matches that pass a second-nearest-neighbor test [1] with a threshold of 0.7. Inliers and outliers can be easily determined based on a distance threshold of 10 pixels because images in this dataset were collected from a static viewpoint.

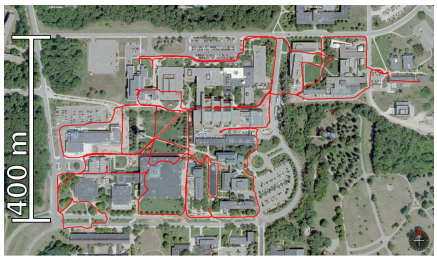
In Fig. 8 we consider the percentage of pairs that could be registered with a minimum of 15 inliers (essentially a practical "bare-minimum" to reliably compute an estimate of camera motion). We see that, in the most challenging region, around 12 h between image pairs, the learned feature descriptors (CMLP and MLP) successfully match over 40% of possible pairs. DAISY performed ever-so-slightly better in this region matching just under 45%. SIFT and SURF match significantly fewer pairs, about 30%. Similar patterns were observed with other minimum inlier thresholds, though the percentage of matches decreases significantly as the minimum required number of inliers increases.

Note that Fig. 8 is smooth because it averages many different pairs, from different locations, with different starting points throughout the day. With smaller sample sizes the relationship between matching and time between images can vary dramatically, i.e., two images collected 8 hours apart at night might match easily, while two images collected 1 hour apart before and after sunset will not be matched.

B. North Campus Long-Term Dataset

One caveat of the previous experiment is that the webcam imagery was taken from a static viewpoint. In order to evaluate the feature descriptors in a more realistic setting we consider their performance on imagery collected by a robotic platform. The imagery was collected in 27 sessions over the course of 15 months on University of Michigan's North Campus (Fig. 9). This dataset contains a wide variety of lighting conditions ranging from early morning to just after dusk. Additionally,

(a) Segway Robot



(b) Sample Session Trajectory

Fig. 9: North Campus Long-Term dataset. A Segway robotic platform (a) was used to collect imagery in University of Michigan's North Campus. In total, 27 sessions (b) were captured over the course of 15 months including a wide variety of lighting conditions and other challenges including seasonal changes, dynamic objects, and construction.

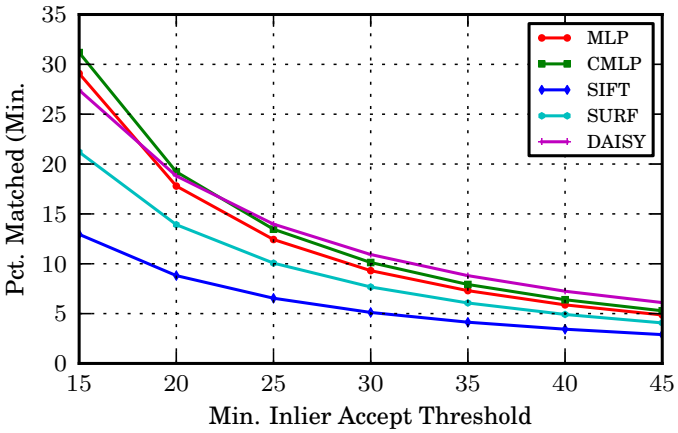
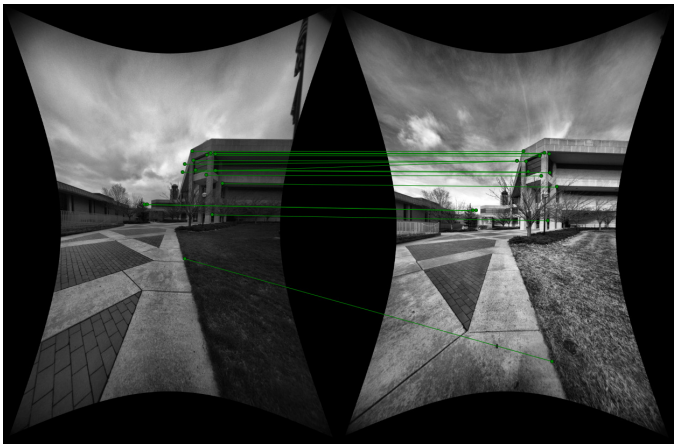


Fig. 10: Average matching results over 500 locations in the North Campus dataset.



(a) Sample Image Registration

(b) Sample Matching Patches

Fig. 11: Sample matching pair that was successfully registered by the CMLP descriptor, but not SIFT, SURF, nor DAISY.

this data includes viewpoint variance and additional challenges caused by moving objects, seasonal changes, and even construction projects. Given known robot pose, the dataset is split up into 500 locations with an average of 37 images per location.

At each location we match all pairs of images. As before, when matching features we perform nearest neighbor matching based on Euclidean distance and employ the second-nearest-neighbor test with a threshold of 0.7. Outliers are rejected by fitting an Essential matrix using random sample consensus [41].

In Fig. 10 we show the percentage of image pairs successfully matched as a function of the minimum number of inliers. Here, we see that again CMLP, MLP, and DAISY provide the best results, matching around 30% of possible pairs at the lowest threshold. SIFT and SURF are significantly less successful. An example image pair that was successfully registered using the CMLP descriptor, but not SIFT, SURF, nor DAISY is shown in Fig. 11.

C. Computation Time

Finally, we provide the computation time of the learned features in Table I. The learned descriptors were developed using Theano and therefore can be computed using the CPU or GPU. For SIFT and SURF we evaluated with OpenCV's CPU version and timing information is provided only as a rough comparison—well optimized GPU versions of both are readily available.

TABLE I: Mean Feature Extraction Time

	CPU	GPU
MLP	0.68 ms/feature	0.07 ms/feature
CMLP	1.34 ms/feature	0.27 ms/feature
SIFT	0.64 ms/feature	—
SURF	0.20 ms/feature	—
DAISY	0.65 ms/feature	—

VI. Discussion and Future Work

Selecting the model parameters for the feature descriptors presents a large number of design choices. This includes the number of layers, the type and dimension of each layer, the activation function, the type of pooling, etc. We feel that the two models used in this work are reasonable and good representatives of two points in the configuration space. However, many of the other model variations considered during development produced very similar results—a more thorough evaluation of the model choices with respect to performance and complexity would be beneficial.

Additionally, we would like to more thoroughly evaluate some of the other algorithm design choices, including the effect of output dimension and the effect of the smooth similarity measure and its time constant.

Beyond lighting invariance, we believe that a similar training scheme could be used in many applications to learn domain specific features. Specifically, we plan to apply the method to underwater imagery in future work.

Finally, it would be beneficial to compare the learned descriptors against additional existing feature descriptors beyond SIFT, SURF, and DAISY.

VII. Conclusions

In this paper, we have presented a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features. We demonstrated that, by tracking feature points in time-lapse videos, one can generate training data that captures how the visual appearance of interest points changes with lighting over time. With this training data we learned feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where Euclidean distance provides good discrimination between matching and non-matching image patches. The learned features provided better image registration performance on a challenging robotic dataset than hand-designed features including SIFT and SURF.

References

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[3] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, St. Louis, MO, Oct. 2009, pp. 1156–1163.

[4] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1645–1661, 2013.

[5] E. Johns and G.-Z. Yang, "Feature co-occurrence maps: Appearance-based localisation throughout the day," in *Proc. IEEE Int. Conf. Robot. and Automation*, Karlsruhe, Germany, May 2013, pp. 3212–3218.

[6] ——, "Generative methods for long-term place recognition in dynamic scenes," *Int. J. Comput. Vis.*, vol. 106, no. 3, pp. 297–314, 2013.

[7] P. Neubert, N. Sunderhauf, and P. Protzel, "Appearance change prediction for long-term navigation across seasons," in *Proc. European Conf. Mobile Robotics*, Barcelona, Spain, Sep. 2013, pp. 198–203.

[8] H. Lategahn, J. Beck, B. Kitt, and C. Stiller, "How to learn an illumination robust image feature for place recognition," in *IEEE Intel. Vehicles Symp.*, Gold Coast, Australia, Jun. 2013, pp. 285–291.

[9] M. Milford, E. Vig, W. Scheirer, and D. Cox, "Towards condition-invariant, top-down visual place recognition," in *Australasian Conference on Robotics and Automation*, Sydney, Australia, Dec. 2013, pp. 1–10.

[10] A. Ranganathan, S. Matsumoto, and D. Ilstrup, "Towards illumination invariance for visual localization," in *Proc. IEEE Int. Conf. Robot. and Automation*, Karlsruhe, Germany, May 2013, pp. 3791–3798.

[11] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2, Nice, France, Oct. 2003, pp. 1470–1477.

[12] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, vol. 2, 2006, pp. 2161–2168.

[13] B. Babenko, P. Dollar, and S. Belongie, "Task specic local region matching," in *Proc. IEEE Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[14] G. Hua, M. Brown, and S. Winder, "Discriminant embedding for local image descriptors," in *Proc. IEEE Int. Conf. Comput. Vis.*, Rio de Janeiro, Brazil, Oct. 2007, pp. 1–8.

[15] S. A. J. Winder and M. Brown, "Learning local image descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Minneapolis, MN, 2007, pp. 1–8.

[16] S. Winder, G. Hua, and M. Brown, "Picking the best DAISY," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Miami, FL, Jun. 2009, pp. 178–185.

[17] M. Brown, G. Hua, and S. Winder, "Discriminative learning of local image descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 43–57, 2010.

[18] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit, "Boosting binary keypoint descriptors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Portland, OR, Jun. 2013, pp. 2874–2881.

[19] E. Tola, V. Lepetit, and P. Fua, "DAISY: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, 2010.

[20] D. Stavens and S. Thrun, "Unsupervised learning of invariant features using video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, San Francisco, CA, Jun. 2010, pp. 1649–1656.

[21] A. Mikulik, M. Perdoch, O. Chum, and J. Matas, "Learning a fine vocabulary," in *Proc. European Conf. Comput. Vis.*, Hersonissos, Greece, Sep. 2010, pp. 1–14.

[22] J. Philbin, M. Isard, J. Sivic, and A. Zisserman, "Descriptor learning for efficient retrieval," in *Proc. European Conf. Comput. Vis.*, Hersonissos, Greece, Sep. 2010, pp. 677–691.

[23] G. Shakhnarovich, "Learning task-specific similarity," Ph.D. dissertation, Massachusetts Inst. Tech., Sep. 2005.

[24] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah, "Signature verification using a 'Siamese' time delay neural network," *Int. J. Pattern Recog. and Artifical Intell.*, vol. 7, no. 4, pp. 669–688, 1993.

[25] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verication," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, San Diego, CA, Jun. 2005, pp. 539–546.

[26] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, New York, NY, Jun. 2006, pp. 1735–1742.

[27] H. Mobahi, R. Collobert, and J. Weston, "Deep learning from temporal coherence in video," in *Proc. Int. Conf. on Machine Learn.*, Montreal, Canada, Jun. 2009, pp. 737–744.

[28] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus, "Learning invariance through imitation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Providence, RI, Jun. 2011, pp. 2729–2736.

[29] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[30] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. of Appr. Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[31] ——, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. of the Int. Conf. on Artif. Intell. and Stats.*, San Juan, Puerto Rico, Mar. 2007, pp. 412–419.

[32] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Proc. Advances Neural Inform. Process. Syst. Conf.*, Whistler, Canada, Dec. 2004, pp. 513–520.

[33] G. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Proc. Advances Neural Inform. Process. Syst. Conf.*, Vancover, Canada, Dec. 2002, pp. 1–8.

[34] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[35] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. Int. Conf. on Machine Learn.*, Haifa, Israel, Jun. 2010, pp. 807–814.

[36] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu, "Deep learning of invariant features via simulated fixations in video," in *Proc. Advances Neural Inform. Process. Syst. Conf.*, Lake Tahoe, NV, Dec. 2012, pp. 3212–3220.

[37] J. Bergstra and Y. Bengio, "Slow, decorrelated features for pretraining complex cell-like networks," in *Proc. Advances Neural Inform. Process. Syst. Conf.*, Vancover, Canada, Dec. 2009, pp. 99–107.

[38] M. Brown, S. Winder, and G. Hua., "Learning local image descriptors data," http://www.cs.ubc.ca/∼mbrown/patchdata/patchdata.html, 2011.

[39] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 3, pp. 448–461, 2010.

[40] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: A CPU and GPU math compiler in python," in *Proc. of the Python for Sci. Comp. Conf.*, Austin, TX, 2010.

[41] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.