Mid-term Project Report (Group-4)

Ayush Kumar Singh
ai20btech11028@iith.ac.in

Digjoy Nandi
ai20btech11007@iith.ac.in

Vojeswitha Gopireddy
ai20btech11024@iith.ac.in

Omkaradithya R Pujari
ai20btech11017@iith.ac.in

Abstract

Our preliminary report discussed traditional algorithmic approaches such as SIFT and SURF detectors. Now, we will discuss a learning-based approach to the image-matching problem. The literature review of this report has been divided into two sections: classical machine learning-based approach and deep learning-based approach. We further tried implementing some face recognition approaches using the YALE face dataset. We also tried to train a deep learning-based feature descriptor using the Homography patches dataset.

1. Introduction

Local features play a crucial role in many Computer Vision applications. Finding and matching them across images has been the subject of vast research. Until recently, the best techniques relied on carefully hand-crafted features. Over the past few years, as in many areas of Computer Vision, methods based on Machine Learning, specifically Deep Learning, have started to outperform these traditional methods.

As discussed before (PPR), in SIFT, we first detect regions of interest(key points) and then assign orientation and local feature descriptors to these key points. Each of these steps can be done using learning-based approaches. We are going to discuss a few of these learning-based approaches. One major shortcoming in SIFT-based image matching is the computational requirement in the inference time. These problems can be mitigated using learning-based approaches.

In this report, we propose a bag-of-words model with SIFT descriptors to recognize different face images of the YALE face dataset. We use different ML-based models in our pipeline and compare their performance to our previously implemented SIFT-based approach (PPR).

2. Literature Review:

2.1. Classical learning-based detectors:

2.1.1 Machine Learning for high-speed corner detection

A high-speed feature detector is necessary when feature points are used in real-time frame-rate applications. Feature detectors such as SIFT (DoG), Harris and SUSAN are good methods which yield high-quality features. However, they are too computationally intensive for use in real-time applications of any complexity.

2.1.2 FAST: Features from Accelerated Segment Test

The segment test considers a circle of sixteen points around a point, say p, and classifies it as a corner point if there is a set of n contiguous pixels in the circle which are brighter than $I_p + t$ or darker than $I_p - t$, where I_p is the intensity at point p, and t is the threshold. The test initially examines 4 pixels chosen symmetrically from the circle and excludes most non-corner points, thus increasing its performance. However, it doesn't generalise well for $n \leq 12$. Moreover, the choice and ordering of fast test pixels contain implicit assumptions about the distribution of features.

2.1.3 Machine Learning a corner detector

Using machine learning, we address above mentioned two issues of FAST. First, a corner detector is built into the algorithm, which detects the corners using the segment test criterion for given n and threshold t. A slow algorithm which tests all 16 pixels on the circle for a given point is used for the above. For each location on the circle, the pixel at the location can be classified in one of the three states wrt to p, as below

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \leq I_p - t \\ s, & I_p - t \leq I_{p\to x} \leq I_p + t \\ d, & I_p + t \leq I_{p\to x} \end{cases} \tag{1}$$

Thus, for a chosen location x we can compute $S_{p\to x}, \forall p \in P$ (the set all pixels in all the training images) and accordingly partition P into subsets P_d, P_s, P_b.
So, in the second stage of the algorithm, we search for the x, which yields the most information about the corner and non-corner candidate pixels. The information gain is measured as below,

$$H(P) - H(P_d) - H(P_s) - H(P_b) \tag{2}$$

where $H(P)$ is the entropy of P measured from K_p, where K_p is a Boolean variable true if p is a corner. Else false.

$$H(P) = (c + \bar{c})\log_2(c + \bar{c}) - c\log_2 c - \bar{c}\log_2 \bar{c} \tag{3}$$

where $c = |\{p|K_p\text{is true}\}|$, i.e. number of corners in the set P and $\bar{c}$ are the number of non-corners in set P

Having selected the value of x which maximises the information gain, we recursively find for x_b, x_s, x_d which partition P_b, P_s, P_d, such that they yield maximum information about the set it is applied to. The process is continued till we reach a subset whose entropy is zero. This gives us a decision tree which can classify all the corners and embodies to rules of FAST. This decision tree is converted to C-code as a series of if-else statements and used as a corner detector.

2.1.4 Non-maximal suppression

To apply non-maximal suppression, a score function, V, is computed for each detected corner. Then the suppression is applied to remove corners which have an adjacent corner with higher V. Computation of the score V is given by the formulae,

$$V = \max(\sum_{x\in S_{\text{bright}}} |I_{p\to x} - I_p| - t, \sum_{x\in S_{\text{dark}}} |I_p - I_{p\to x}| - t) \tag{4}$$

2.2. Deep learning based detector:

A deep learning-based approach can replace all the steps in traditional image-matching algorithms, such as SIFT. Here we will discuss deep learning-based methods for local feature descriptors in detail. We also look into deep learning-based orientational estimators.

2.2.1 Local feature descriptors using Deep Learning method

Once discriminative interest points are detected from raw images, a local patch descriptor must be coupled for each feature to establish feature correspondence correctly and efficiently across two or more images. In other words, the feature descriptors are commonly used to transform the original local information around the interest point into a stable and discriminative form, usually as a high-dimensional vector, so that two corresponding features are as close as possible in the descriptor space, two non-corresponding features are as far as possible. Due to the nature of the problem, Siamese networks are generally used for finding local image descriptors. Since training Siamese network can be tricky, various improvements, such as different kinds of loss functions and stable learning methods, has been proposed. We will discuss some of those loss functions used in the Siamese network.

Loss functions:

Let $\mathcal{X} = \{x_1, x_2, ..., x_N\}, x_i \in \mathcal{R}^{mxn}$ denote a set of N training patches with m × n pixels. $\{y_ij, i, j < N\}$ is a set of pairwise labels for $\mathcal{X}$ indicating whether x_i and x_j belong to the same class ($y_ij = 1$) or not ($y_ij = 0$). We call the matching pairs with $y_ij = 1$ the matching pairs and those with $y_ij = 0$ the non-matching pairs. The goal of descriptor learning is to learn a feature embedding $f(.) : \mathcal{R}^{mxn} \to \mathcal{R}^d$ that maps raw patch pixels to a d dimensional vector, such that $||f(x_i) - f(x_j)||_2$ is small when $y_ij = 1$ and $||f(x_i) - f(x_j)||_2$ is large when $y_ij = 0$. We assume that $f(.)$ lives on the unit sphere, i.e., $||f(x)||_2 = 1, \forall x \in \mathcal{R}^{mxn}$.

Cross entropy loss

The cross-entropy loss is given by

$$l_{cross} = -\frac{1}{N}\sum_{i=0}^{N} y_i \log\left(f(x_i)\right) + \\ (1 - y_i)\log\left(1 - f(x_i)\right) \tag{5}$$

Here, $f(x_i)$ and $(1 - f(x_i)$ are the Softmax activations, i.e.

$$f(x_i) = -\frac{1}{N}\sum_{i=0}^{N}\log\left(\frac{\exp\left(x_i + b_i\right)}{\sum_{j=0}^{n}\exp\left(x_j + b_j\right)}\right) \tag{6}$$

where b_i is the bias term corresponding to class y of i-th image and b_j is the bias term corresponding to j-th class.

Pairwise loss

The pairwise loss tries to induce a small distance for matching pairs and a large distance for non-matching pairs. An input for pairwise loss is (x_i, x_j, y_ij), consisting of a pair of samples and their corresponding label. The most widely used pairwise loss is the contrastive loss:

$$l_{con} = y_{ij}\max\left(0, ||f(x_i) - f(x_j)||_2 - \epsilon^+\right) \\ (1 - y_{ij})\max\left(0, \epsilon^- - ||f(x_i) - f(x_j)||_2\right) \tag{7}$$

Here $f(.)$ is the feature embedding. ϵ^+ and ϵ^- control the margins of the matching and non-matching pairs, respectively. The main problem with the pairwise loss is that the margin parameters are often difficult to choose.

Triplet loss

Triplet loss takes a triplet of samples as input. One triplet consists of three samples: (x_i, x_j, x_k), with $y_ij = 1$ and $y_ik = 0$. To simplify the notation, we denote one triplet as (x_i, x_i^+, x_i^-), where $x_i^+ = x_j$ and $x_i^- = x_k$. One commonly used triplet loss is the ranking loss.

$$\begin{aligned} l_{tri} = \max(0, \epsilon - ||f(x_i) - f(x_i^-)||_2 \\ - ||f(x_i) - f(x_i^+)||_2) \end{aligned} \tag{8}$$

where ϵ is a margin. The idea of ranking loss is to separate the matching and non-matching samples by at least a margin ϵ.

The main difference between pairwise loss and triplet loss is that pairwise loss considers the absolute distances of the matching pairs and non-matching pairs. In contrast, triplet loss considers the relative distance difference between matching and non-matching pairs. Since the quality of the embeddings largely depends on the relative ordering of the matching and non-matching pairs, triplet loss performs better than the pairwise loss in local descriptor learning.

Pitfalls:

The main issue of triplet loss and pairwise loss is that as the number of training samples grows, sampling all the possible triplets/pairs becomes infeasible, and only a relatively small portion of triplets/pairs can be used in training. The training is often ineffective since many of the sampled triplets/pairs will satisfy the constraint within just a few training steps.

Improvements:

Some possible solutions to improve the loss function are as follows:

- Remove the "easy samples" and add new "hard samples" to the training set. However, determining which samples to remove or add is a challenging task. Additionally, focusing only on the samples that violate the most training constraints will lead to overfitting.
- An improved version of triplet loss by applying in-triplet hard negative mining. The idea is that one triplet contains two non-matching pairs $(x_i, x_i^-$ and $(x_i^+, x_i^-$, and choosing the one that violates the triplet constraint more will make training more effective.
- Use a global loss and combine it with the traditional triplet loss to address the sampling issue in pairwise and triplet loss. Instead of considering a sample pair or triplet, the global loss considers all matching and non-matching pairs in one training batch. It calculates the empirical mean and variance of the distance of the matching and non-matching pairs. The main idea of the global loss is to separate two empirical means by a margin and minimize the variances.
- Use structured loss, which considers s all the possible matching and non-matching pairs in one batch of samples. By carefully designing the loss functions, the structured loss can focus on the "hard" pairs in training.

Spread out local descriptors

To fully utilize the descriptor space, the descriptors should be sufficiently "spread-out" in the descriptor space. Spread-out descriptor space means that given a dataset, we say that the learned descriptors are spread-out if two randomly sampled non-matching descriptors are close to orthogonal with a high probability.

Proposition 1 *Let $\vec{p_1}, \vec{p_2} \in S^{d_1}$ be two points independently and uniformly sampled from the unit sphere in d-dimensional space. A d-dimensional l_2 normalized vector represents each point. The probability density of $\vec{p_1}^T\vec{p_2}$ satisfies*

$$p(\vec{p_1}^T\vec{p_2} = s) = \begin{cases} \dfrac{(1-s^2)^{\frac{d-1}{2}} - 1}{B(\frac{d-1}{2}, \frac{1}{2})} & -1 \le s \le 1 \\ 0 & \textit{otherwise} \end{cases} \tag{9}$$

where B(a,b) is the beta function.

The probability distribution of the two points' inner product (cosine similarity) is independently and uniformly sampled from the unit sphere. It shows that with high probability, the two independently and uniformly sampled points are close to orthogonal.

Proposition 2 *Let $\vec{p_1}, \vec{p_2} \in S^{d1}$ be two points independently and uniformly sampled from the unit sphere. The mean and the second moment of $\vec{p_1}^T\vec{p_2}$ are*

$$\mathcal{E}(\vec{p_1}^T\vec{p_2}) = 0 \tag{10}$$

$$\mathcal{E}((\vec{p_1}^T\vec{p_2})^2) = \frac{1}{d} \tag{11}$$

*Global orthogonal regularization Global orthogonal regularization tries to match the mean and the second moment shown in Proposition 2. It encourages that the descriptors of randomly sampled nonmatching pairs have similar

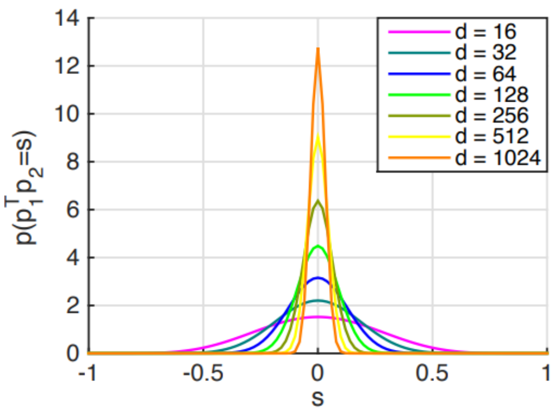


Figure 1. Probability density of the inner product of two points

statistical properties as two points independently and uniformly sampled from the unit sphere. The sample mean of the inner product of the descriptors of non-matching pairs is,

$$M_1 = \frac{1}{N}\sum_{i=1}^{N} f(x_i)^T f(x_i^-) \tag{12}$$

The sample second moment of the inner product is,

$$M_2(f(x)^T f(x^-)) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i)^T f(x_i^-))^2 \tag{13}$$

The Global Orthogonal Regularization (GOR) is defined as

$$l_{gor} = M_1^2 + \max(0, M_2 - \frac{1}{d}) \tag{14}$$

where d is the dimension of the final output descriptor.

The first term tries to match the mean of the distributions, and the second term tries to make the second moment close to $1/d$. In practice, we use a sampled batch to estimate its value. The reason for using the hinge loss for the second term is that, in many batches, all the non-matching pairs are already very close to being orthogonal ($M_2 < 1/d$), and there is no need to force M_2 to be $1/d$.

The proposed regularization term can be used with any loss function. Denote the original training loss as $l_{()}$, the final loss can be written as,

$$l_{(.)-gor} = l_{(.)} + _{gor}, \tag{15}$$

where α is a tunable parameter.

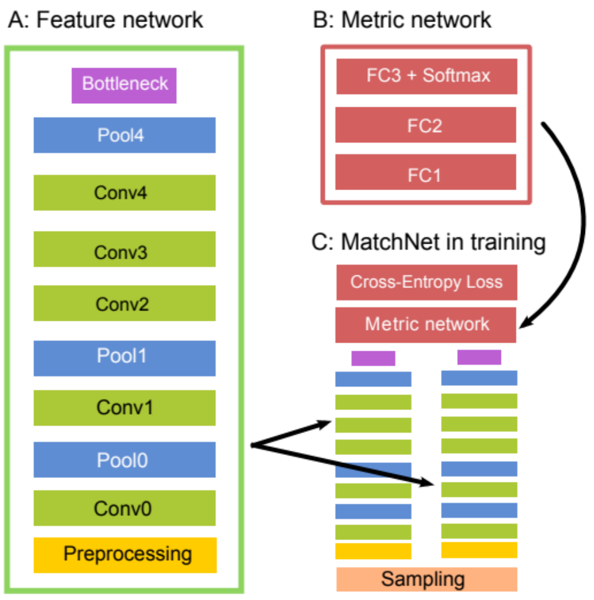


Figure 2. MatchNet architecture

2.2.2 MatchNet

One example of a deep learning method for local feature learning is MatchNet. MatchNet is a deep-network architecture for jointly learning a feature network that maps a patch to a feature representation and a metric network that maps pairs of features to a similarity. However, unlike in SIFT, where two descriptors are compared in feature space using the Euclidean distance, MatchNet compares the representations using a learned distance metric implemented as a set of fully connected layers.

Two-tower structure with tied parameters

During training, we employed two feature networks (or "towers") that connect to a comparison network, with the constraint that the two towers share the same parameters. Updates for either tower will be applied to the shared coefficients. This approach is related to the Siamese network and minimized using cross-entropy loss.

2.2.3 Orientation Assignment to feature points

The orientation of the feature vector is decided through various techniques depending on the detector. However, the ideal canonical orientation is undefined. Given the patch around a feature point, the proposed method allows learning optimization to find the most reliable and effective orientation.

Formulation

The orientation of the feature vector is learnt as an implicit variable by minimising the descriptor distances of pairs of local features corresponding to the same physical point. Siamese networks are used for learning the orientation whose loss function is formulated as $\sum_i \mathcal{L}(p_i)$ over the parameters W of the network, with

$$\mathcal{L}(p_i) = ||g(p_i^1, f_w(p_i^1)) - g(p_i^2, f_w(p_i^2))||_2^2 \tag{16}$$

where the pairs $p_i = \{p_i^1, p_i^2\}$ are pairs of image patches from training set.
The CNN is trained to predict two values, scaled as sine and cosine, to alleviate the periodicity problem while calculating the angle. Thus angle is calculated as,

$$f_w(p_i^*) = arctan2(\hat{f}_w^1(p_i^*), \hat{f}_w^2(p_i^*)) \tag{17}$$

Generalized Hinging Hyperplane Activation

A GHH activation function is employed in the network layers of CNN. The GHH activation function is a generalization of ReLU, max out, and the PReLU activation functions based on Generalised Hinging Hyperplanes. For a given output layer, $y = \{y_{1,1}, y_{1,2}, ..., y_{2,1}, ..., y_{S,M}\}$ before activation, activation function is defined as,

$$o(y) = \sum_{s \in \{1,2,...,S\}} \delta_s \max_{m \in \{1,2,...,M\}} y_{s,m} \tag{18}$$

where

$$\delta_s = \begin{cases} 1 & \text{if s is odd} \\ -1 & \text{otherwise} \end{cases} \tag{19}$$

and S and M are meta-parameters controlling the number of planar segments.

3. Proposed classical machine learning based method

This method combines SIFT features with machine learning models, such as decision trees, SVM, etc., to classify images. The idea here is to use bag-of-visual words to calculate image features. We first extract local features from training images. We form clusters of these features using any clustering algorithm (kNN in our case). Once we have found the cluster centres, we quantize the local features of our training images using these points. We then train our ML model on these features.

3.1. Experimental Results:

We have used a subset of Yale face data for benchmarking results. It contains 165 images for 15 subjects, with 11 images/per person. The images contain different facial expressions and illumination conditions for each subject. The image size is 243 × 320 pixels. Figure 5 shows a sample of images from this database. The raw faces were used without preprocessing (cropping, normalization, histogram equalization, etc.) to assess the robustness of the algorithms in the comparison.

Clusters	Accuracy	Method
50	77.77	SVM + SIFT
50	64.44	XGBoost + SIFT
80	79.23	SVM + SIFT
80	67.23	XGBoost + SIFT
100	86.66	SVM + SIFT
100	71.11	XGBoost + SIFT
200	84.44	SVM + SIFT
200	75.55	XGBoost + SIFT

Table 1. Cluster vs accuracy

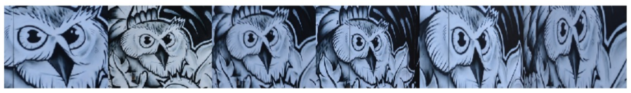


Figure 3. Example image sequence

3.1.1 Training:

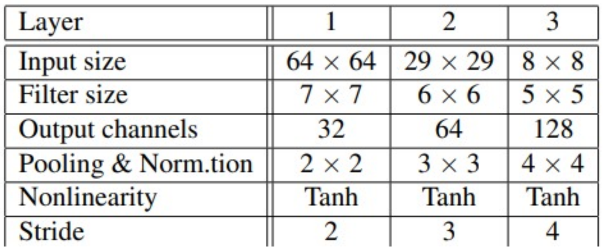
We divided our dataset into 120 training images and 45 testing images. Each subject had 8 images in the training dataset and 3 in the testing dataset. We experimented with different cluster numbers while training. Our best model gave a comparable performance to SIFT-based face recognition (PPR) with a much less inference time.

4. Proposed deep learning-based feature descriptors

For training, we have used the Homography patch dataset. In this dataset, there are 116 image sequences wherein each image sequence contains a reference image and 5 target images taken under different illumination and different viewpoints.

Patches are sampled in the reference image using a combination of local feature extractors (Hessian, Harris and DoG detector). Lowe's method estimates the patch orientation using a single major orientation. No affine adaptation is used. Therefore, all patches are square regions in the reference image.

Patches are extracted from regions with a scale magnified by a factor of 5 compared to the original detected fea-



Layer	1	2	3
Input size	64×64	29×29	8×8
Filter size	7×7	6×6	5×5
Output channels	32	64	128
Pooling & Norm.tion	2×2	3×3	4×4
Nonlinearity	Tanh	Tanh	Tanh
Stride	2	3	4

Figure 4. Architecture of the proposed 3-layer network.

Figure 5. Example image patches

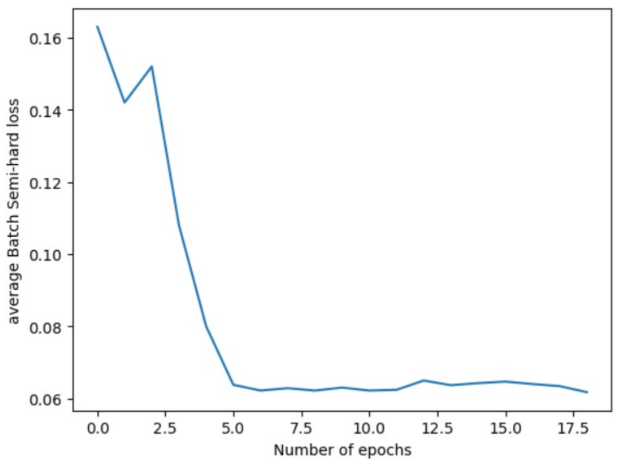


Figure 6. Testing loss vs Epoch

ture scale. Only patches for which this region is entirely contained in the image are kept. To simulate the geometric repeatability error of a local feature detector, affine jitter is applied to the patches. Based on the amount of jitter applied, patches can be divided into easy and hard patches. Due to computational limitations, we have only used a single image sequence for training and testing.

4.1. Training:

We used the triplet loss function to train our CNN-based Siamese network. While training, it was found that random triplet sampling does not improve the model after only a few steps. Therefore, we have used semi-hard mining with a margin of 0.2 to train our network.

We have used a batch size of 1000 images while training. We use one of the noisy patch images for testing. ADAM optimizer with a learning rate of 0.001 was used. Our testing loss started oscillating after a few epochs, which can be improved with careful supervision.

5. Future Work:

Our aim for the final report is to make a complete pipeline for image matching. We aim to improve the implementation of local feature detectors by building upon this report. We also aim to create a robust and efficient performance of orientation learning.

6. Conclusion

We discussed various learning-based data-driven approaches for image matching. We looked at various deep learning-based approaches to train local descriptors in depth and tried implementing a simple model to learn feature descriptors. We also looked at different loss functions involved in such methods and tried replicating that in our implementation. A classical-based model combined with SIFT was also introduced, which gave comparable results to SIFT based approach with much less inference time.

7. References:

1. Han, X., Leung, T., Jia, Y., Sukthankar, R., Berg, A.C.: MatchNet: Unifying Feature and Metric Learning for Patch-Based Matching. In: CVPR. (2015)

2. Yi, K., Verdie, Y., Lepetit, V., Fua, P.: Learning to Assign Orientations to Feature Points. In: CVPR. (2016)

3. Zagoruyko, S., Komodakis, N.: Learning to Compare Image Patches via Convolutional Neural Networks. In: CVPR. (2015)

4. LIFT: Learned Invariant Feature Transform Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, Pascal Fua

5. Edward Rosten and Tom Drummond, "Machine learning for high speed corner detection" in 9th European Conference on Computer Vision, vol. 1, 2006, pp. 430–443.

Preliminary Report: PDF
Code Link: Kaggle notebook 1, Kaggle notebook 2