Preliminary 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

This project aims to develop a robust and efficient image-matching algorithm for various applications in computer vision. The project can be divided into two stages where we first discuss algorithmic approaches for image matching, and the second is the deep learning-based approach. In this preliminary report, we have discussed a few traditional approaches for feature extraction, such as SURF and SIFT. We then used these extracted features for the image-matching task. To test these approaches, we have used one of them(SIFT) for the face recognition task.

1. Introduction

Image matching is a technique used in computer vision to identify and compare two or more images to determine if they are the same or similar. It involves finding corresponding points or features in two or more images and then using algorithms to compare them.

There are various applications of image matching, including object recognition, facial recognition, and image retrieval. For example, in object recognition, an algorithm might be trained to identify a specific object, such as a car, and then use image matching to locate that object in different images or video streams.

Image matching is a complex process that involves several steps, such as feature extraction, feature matching, and geometric verification. Feature extraction involves identifying and extracting distinctive points or features from an image, such as corners, edges, or blobs. Feature matching involves finding corresponding features between two or more images. Geometric verification involves determining the transformation between two images, such as rotation, translation, and scaling, to align them.

In this report, we explore various feature-extracting algorithm used for image matching.

2. Literature Review

2.1. SIFT

Scale Invariant Feature Transform algorithm has been proposed for extracting stable features that are invariant to rotation, scaling and partially invariant to changes in illumination and affine transformation from images to perform matching of different views of an object or scene. The resulting features are highly distinctive. Following are the major stages of computation used to generate the set of image features

2.1.1 Keypoint Localization in scale space

SIFT method uses scale-space Difference-of-Gaussian (DoG) to detect interest points in images. Let $I(x, y)$ is an input image, $L(x, y, \sigma)$ is the scale space defined as a function

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

where $G(x, y, \sigma)$ is the Gaussian function

The Difference-of-Gaussian function $D(x, y, \sigma)$ can be generated by subtracting each image from its direct neighbours with a multiplicative factor k given by:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned} \tag{2}$$

The are multiple reasons to use this function. It is easy to compute and provides a good approximation of scale normalized Laplacian of Gaussian.

2.1.2 Elimination of weak key points

The keypoint candidates are detected by comparing each point to its Eight neighbours on the same scale, and each of its 9 neighbours one scale up and down. Every point, bigger

or smaller than each of its neighbours, is a key point candidate. Keypoint candidates located on edge or with poor contrast are eliminated. This is done to avoid unstable key points in case of noisy images.

2.1.3 Orientation Assignment

By assigning a consistent orientation to each key point based on local image properties, the keypoint descriptor can be represented relative to this orientation and achieve invariance to image rotation. The gradient magnitude m(x,y) and orientation $\theta(x, y)$ are computed using pixel differences in the equation below. Then an orientation histogram with 36 bins is computed from the image gradients around the key point. The maximum orientation is assigned to this key point. For each other orientation within 80% of the maximum orientation, a new key point with this orientation is created. Each key point is rotated in the direction of its orientation and then normalized.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad (3)$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x+1,y) - L(x-1,y))}{(L(x,y+1) - L(x,y-1))}\right) \quad (4)$$

2.1.4 Local Image Descriptor

The next step is to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations, such as changes in illumination or 3D viewpoint. The area around the key point is divided into 4*4 subregions. Then an orientation histogram with eight bins is built for each subregion, and a Gaussian window weights the corresponding gradient values. This results in a vector with 128 dimensions (4 * 4 * 8). The vector is normalized to unit length, which grants invariance to multiplicative changes in lighting.

2.2. Harris Detector

Harris Corner Detector is a corner detection operator commonly used in computer vision algorithms to extract corners and infer features of an image. Corners are the essential features in the image, and they have generally termed interest points invariant to translation, rotation, and illumination.

2.2.1 Corner detection

The idea is to consider a small window around each pixel p in an image. We take the sum squared difference (SSD) of the pixel values before and after the shift and identify

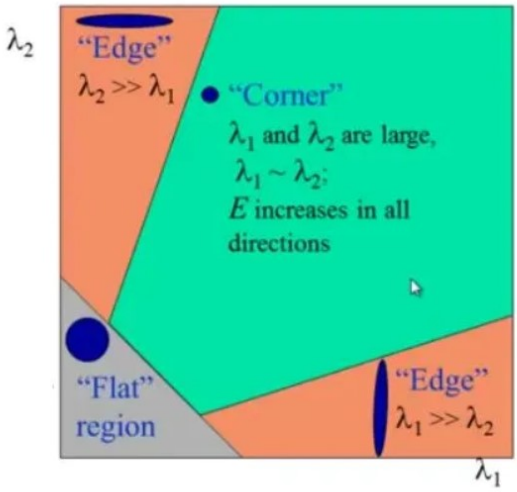

Figure 1. Corner detection using Harris detector

pixel windows where the SSD is large for shifts in all eight directions. Let us define the change function E(u,v) as the sum of all the sum squared differences (SSD), where u,v are the x,y coordinates of every pixel in our 3 x 3 window, and I is the intensity value of the pixel. The image's features are all pixels with large values of E(u,v), as defined by some threshold.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad (5)$$

We must maximize this function E(u,v) for corner detection. That means we have to maximize the second term. Applying Taylor Expansion to the above equation and using some mathematical steps, we get the final equation:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \quad (6)$$

where $M = \sum w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ By solving for the M eigenvectors, we can obtain the directions for the largest and smallest increases in SSD. The corresponding eigenvalues give us the actual value amount of these increases. A score, R, is calculated for each window:

$$R = det(M) - k(tr(M))^2 \quad (7)$$

$$det(M) = \lambda_1 \lambda_2 \quad (8)$$

$$tr(M) = \lambda_1 + \lambda_2 \quad (9)$$

Here, λ_1 and λ_2 are eigenvalues of M. So the values of these eigenvalues decide whether a region is a corner, edge or flat shown in figure 1.

- When R is small, which happens when λ_1 and λ_2 are small, the region is flat.
- When $R < 0$, which happens when $\lambda_1 >> \lambda_2$ or vice versa, the region is an edge.
- When R is large, which happens when λ_1 and λ_2 are large and $\lambda_1 \approx \lambda_2$, the region is a corner.

2.3. SURF Detector

Speeded-Up Robust Features (SURF) is a scale and rotation invariant detector and descriptor, relying on integrated images for detection and matching steps. To enhance the computational speeds of the SURF without compromising performance, we need to simplify the detection schema while being accurate and ensure the distinctiveness of the descriptor while decreasing its size.

2.3.1 Integral Images

SURF uses Hessian-Matrix approximation for localising key points, whose computation can be reduced using Integral images. They allow faster computation of convolution filters independent of filter size. Let $I(x, y), I_{\sum}(x, y)$ denote the input image and integral image, respectively. We have,

$$I_{\sum}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{10}$$

2.3.2 Keypoint Detection using Hessian matrix

SURF computes the determinant of the Hessian matrix to detect key points and also for scale selection. The Hessian matrix $\mathcal{H}(x, y, \sigma)$ in (x, y) at scale σ is defined as,

$$\mathcal{H}(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \tag{11}$$

where $L_{xx}(x, y, \sigma)$ is the convolution of Gaussian second order partial derivative wrt image I at (x, y). The approximate Gaussian derivatives can be computed very cheaply and independent of filter size.
Let D_{xx}, D_{xy}, D_{yy} be the approximation for the second-order partial derivatives. Then we have

$$det(\mathcal{H}_{\text{approx}}) = D_{xx}D_{yy} - (wD_{xy})^2 \tag{12}$$

where w is the relative weight defined as,

$$w = \frac{|L_{xy}(\sigma)|_F |D_{yy}(\sigma)|_F}{|L_{yy}(\sigma)|_F |D_{xy}(\sigma)|_F} \tag{13}$$

where $|x|_F$ is the Frobenious norm. In practice, the relative weight w is taken to be a constant.

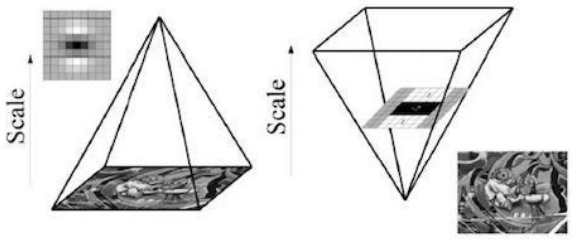

Figure 2. Scale Space Pyramid

2.4. Scale Space Representation

In SURF, scale-spaces are implemented as an image pyramid. Differences in Gaussians (DoG) images, which identify edges and blobs, can be built using differences in the pyramid layers. The images are repeatedly smoothed and sub-sampled to achieve a higher pyramid level. The scale size is analysed by up-scaling the filter size rather than reducing the image size. Since we use box filters and integral images, we can apply any filter to the original image at the same speed.
Scale spaces are divided into octaves. For up-scaling, the filters at constant cost, the filter size and sampling intervals for extracting key points are doubled simultaneously.

2.5. Keypoint Description and Matching

2.5.1 Orientation Assignment

For the SURF detector to image orientation invariant, Haar wavelet responses are calculated in x and y direction within a circular neighbourhood of radius $6s$ around the keypoint, where s is the scale. The size of wavelets is also scale dependent and is set to $4s$ side length. For fast filtering, integral images are used.
After calculating and weighing the wavelet responses, the dominant orientation is calculated as the sum of all responses within a sliding orientation window of size $\frac{\pi}{3}$ The summed horizontal and vertical responses yield the local orientation vector, longest of which defines the orientation of the keypoint.

2.5.2 Descriptor Components

We construct a square region around the key point oriented along its orientation to extract the descriptor. As shown in 3, we split the region into smaller 4×4, and the Haar wavelet responses are calculated for each sub-region. Let d_x, d_y be the Haar wavelet responses in the x and y directions; the four-dimensional descriptor vector v for its intensity struc-

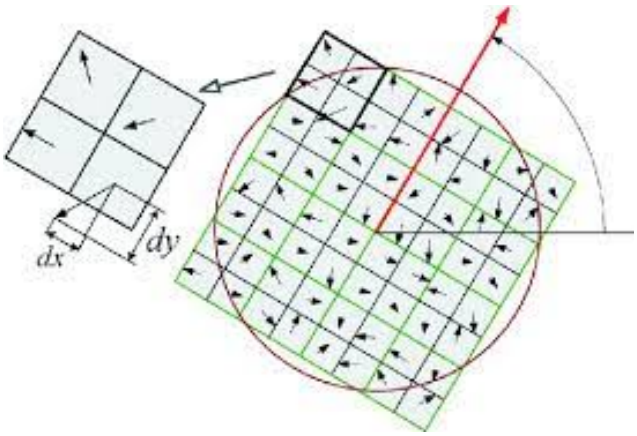

Figure 3. Oriented quadratic grids of the descriptor

ture is defined for each sub-region as below,

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \qquad (14)$$

3. Experiments:

By using image matching, we can compare images from different viewpoints, and the same approach can be used for several computer vision applications. Face recognition is one task where image matching can be used to compare the same object(face). 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.

3.1. Approach

We have used SIFT local features for image matching. Given a new face image, the features extracted from that face are compared against the features from each face in the database. The face in the database with the most significant number of matching points is considered the nearest face and is used to classify the new face. A feature is considered matched with another feature when the distance to that feature is less than a specific fraction of the distance to the next nearest feature. We have experimented with different ratios to get optimal results.

3.2. Results

We found a range between 0.6 and 0.8 optimal when using different threshold values. The results are shown in table Table:1.

Ratio Threshold	Accuracy	Ratio Threshold	Accuracy
0.9	79.39	0.6	90
0.8	86.66	0.5	89.09
0.7	90.3	0.3	83.63

Table 1. Ratio Threshold vs Accuracy

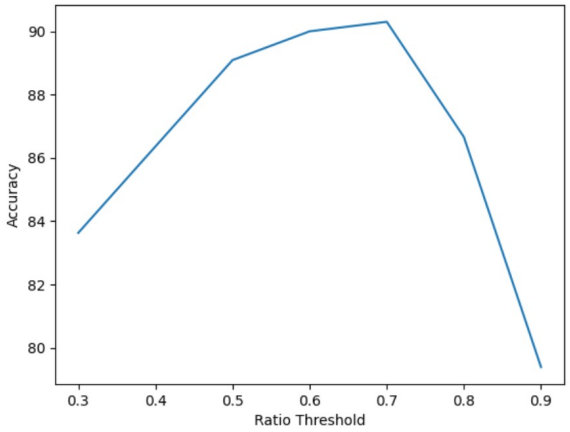

Figure 4. Ratio Threshold vs Accuracy

Figure 5. Feature Extraction using SIFT

Code Link: Kaggle notebook

4. Conclusion

We discussed different algorithmic approaches for image matching. Harris detector detects corners and infers image features, while SIFT calculates highly distinctive invariant features.SURF, similar to SIFT, is a more robust and computationally efficient algorithm. We used SIFT for the face recognition task on a small data set by matching local features, which gave us benchmark results.

5. References

1. D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” Int. J. Comput. Vis., vol. 60, no.2, pp. 91–110, Nov. 2004

2. B. Herbert, E. Andreas, T. Tinne, and G. Luc Van, “Speeded-Up Robust Features (SURF),” Comput. Vis.

Image Understand., vol. 110, no. 3, pp. 346–359, Jun. 2008.

3. M. Aly, “Face Recognition using SIFT Features”,

4. A. Majumdar, R. K. Ward, “Discriminative SIFT Features for Face Recognition,” Department of Electrical and Computer Engineering, University of British Columbia

5. ”A combined corner and edge detector” by Chris Harris Mike Stephens