

4 Feature-based registration

IMAGE REGISTRATION is the process of transforming one image (the moving image) to the coordinate system of another image (the target image). We will compute this transformation by matching image features. Here we will use SIFT features (Scale invariant feature transform) that are described in detail in¹. The reason for using SIFT is that OpenCV² has a good implementation of the SIFT features, which makes it easy to carry out the exercise on feature-based registration. But recent developments on learning-based methods for feature-matching will generally give higher accuracy and faster computation³. Therefore, learning-based methods will generally be a preferred choice. Learning-based methods roughly use the same components as traditional methods like SIFT, which are important for image registration. Traditional interest point features that do not involve learning such as SIFT are referred to as hand-crafted features.

The concept of interest point image features is essential in image analysis. The basic idea is to identify salient positions in an image (unique key points) and use the surrounding image context to describe the image locally in the form of a descriptor vector. This is the same concept as we used for feature-based segmentation, where we wanted to group parts of the image with similar appearance or texture. But in image matching, we are interested in finding positions in two images of the same object or scene that can uniquely be matched.

This principle has been used for solving many problems including object recognition, image retrieval (image search), image stitching, geometric reconstruction, etc. Solutions to many of these problems have however been improved by machine learning methods and especially convolutional neural networks (CNNs) have replaced feature-based analysis in a range of applications. But interest point features are the best performing technique for finding correspondence between images with large structural variation, which is typical for photographs of the same scene or images from a microscope.

Image registration based on interest point features is based on detecting interest points in an image and finding unique correspondence

¹ David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004

² OpenCV. Open source computer vision library, 2015

³ Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129(1):23–79, 2021

between these interest points. Interest point features are composed of two elements, namely an interest point or a key point, which is a position in the images (an (x, y) -coordinate) and a descriptor, which is a vector that encodes the appearance (pixel intensities) in a local neighborhood around the interest point. Typically, hundreds to thousands of features are detected in an image, and correspondence is found by finding the descriptors with the highest similarity between two images.

The desired properties for interest point features that they are

- unique such that they encode just one position in an image,
- invariant to change in view-point, overall intensity change, and change in noise level,
- robust such that they will be detected despite image changes, and
- efficient with respect to memory and computational time.

SIFT has been designed to have exactly these properties, and later developments are modifications that strive at optimizing these properties either through design or machine learning.

Learning-based features Some deep learning-based alternatives to hand-crafted features have been suggested⁴, showing superior performance. High-performance examples include SuperPoint⁵ that learns both interest point detection and description and Key.Net⁶ that only learns interest point detection.

The problem of learning interest point features is that there is no ground truth for what a good interest point is. The SuperPoint method solves this by generating synthetic data, where specific image features such as corners are created and used for training an interest point detector. This detector is then used for detecting interest points in a set of real images. A set of corresponding images are created by using homographies to transform one image and detecting interest points in each of these. Since the homography gives ground truth correspondence, the sum of all detected interest points is used as ground truth. A second model is then trained for detecting these interest points, and the same network is used to simultaneously learn descriptors at the interest point locations. The resulting interest points have the same properties as hand-crafted features, namely a location and a vector that describes the local image appearance, which can then be matched by comparing interest points. The advantage is faster computation and higher accuracy in matching.

Besides learning models for computing interest point features, there are also methods for learning the matching between interest point features, e.g. the SuperGlue method⁷. Also here the advantage is higher performance compared to hand-crafted features.

⁴ Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129(1):23–79, 2021

⁵ Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018

⁶ Axel Barroso-Laguna, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Key.net: Keypoint detection by hand-crafted and learned cnn filters. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5836–5844, 2019

⁷ Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020

Hand-crafted features – SIFT The advantage of hand-crafted features is that there is no learning involved, and therefore they will be easy to use for any type of images. Despite the lower performance compared to learning-based features, hand-crafted features will often perform well for many registration problems.

Scale invariant feature transform (SIFT) described in⁸ is a widely used interest point feature. Here we focus on SIFT. But bear in mind that many other interest point features have similar properties⁹.

We will use SIFT for feature-based image registration. The problem we address is that we are given two images that depict the same object and we want to find an image transformation that aligns the two images. This allows us to compare structures found in one image with structures found in the other image. In some cases, the difference between images can be modeled by a rotation \mathbf{R} , translation \mathbf{t} and scale s . This is e.g. the case in microscopy or CT, where pixels have a fixed physical size. A more general affine transformation can be computed using a homography, but in cases where the imaging system can only transform the image by a rotation, translation, and scale, these will be the appropriate parameters to compute. Here we will work with sets of corresponding 2D points, where the correspondence is found by matching SIFT features.

Fitting two sets of 2D points in least squares sense. For two 2D point sets \mathbf{P} and \mathbf{Q} with corresponding elements \mathbf{p}_i and \mathbf{q}_i (2×1 column vectors), where $i = 1, \dots, n$, we are interested in finding a rotation \mathbf{R} , a translation \mathbf{t} and a scale s that minimizes the squared distance between the two point sets. The transformation from one point set to the other is given by

$$\mathbf{q}_i = s\mathbf{R}\mathbf{p}_i + \mathbf{t}. \quad (4.1)$$

The scale s can be found e.g. by computing the ratio between average distance to the centroid of each point set

$$s = \frac{\sum_{i=1}^n \|\mathbf{q}_i - \mu_q\|}{\sum_{i=1}^n \|\mathbf{p}_i - \mu_p\|}, \quad (4.2)$$

where $\mu_p = \frac{1}{n} \sum_{i=1}^n \mathbf{p}_i$ and $\mu_q = \frac{1}{n} \sum_{i=1}^n \mathbf{q}_i$ are the centroids of \mathbf{p}_i and \mathbf{q}_i respectively.

Now we want to find the rotation and translation that minimize

$$\sum_{i=1}^n (\mathbf{q}_i - s\mathbf{R}\mathbf{p}_i - \mathbf{t})^2.$$

One way of least-squares fitting 2D point sets involves 2-by-2 covariance matrix (normalization is not needed)

$$\mathbf{C} = \sum_{i=1}^n (\mathbf{q}_i - \mu_q)(\mathbf{p}_i - \mu_p)^T,$$

⁸ David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004

⁹ Tinne Tuytelaars, Krystian Mikolajczyk, et al. Local invariant feature detectors: a survey. *Foundations and trends® in computer graphics and vision*, 3(3):177–280, 2008

and its singular value decomposition

$$\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{C}.$$

Here, \mathbf{U} and \mathbf{V} are the left and right singular matrices respectively and Σ is a diagonal 2-by-2 matrix containing the singular values. From this, we obtain the rotation

$$\hat{\mathbf{R}} = \mathbf{U}\mathbf{V}^T. \quad (4.3)$$

In rare cases, this computation can result in a reflection instead of a rotation. If the determinant of $\det(\hat{\mathbf{R}}) = 1$ it is a rotation and if $\det(\hat{\mathbf{R}}) = -1$ it is a reflection. This computation will typically only result in a reflection e.g. if there are only two points for determining the rotation. Therefore, we can compute the rotation taking this into account by

$$\mathbf{R} = \hat{\mathbf{R}}\mathbf{D}, \quad (4.4)$$

where

$$\mathbf{D} = \begin{bmatrix} 1 & 0 \\ 0 & \det(\hat{\mathbf{R}}) \end{bmatrix}. \quad (4.5)$$

Finally, we find the translation as the average vector from points in \mathbf{q} to the rotated points in \mathbf{p}

$$\mathbf{t} = \frac{1}{n} \sum_{i=1}^n (\mathbf{q}_i - s\mathbf{R}\mathbf{p}_i) = \mu_q - s\mathbf{R}\mu_p. \quad (4.6)$$

See e.g. Arun et al.¹⁰ that covers a more general 3D case.

4.1 Exercise on feature-based registration

This exercise aims at finding correspondence between images by matching SIFT features and computing the transformation, i.e. the rotation, translation, and scale between the two images. For computing the SIFT features you can use vlFeat for MATLAB or OpenCV for Python (you will need a newer version of OpenCV). In the extra exercise, you can use the transformation obtained here to compare the fiber diameters that you got from using blob detection for fibers. But for now, the purpose is to compute the transformation.

4.1.1 Rotation, translation and scale

You should implement a function that takes two point sets as input and returns the rotation, translation, and scale. To ensure that you have the correct implementation, you can make a set of random 2D points and ensure that you get the correct numbers out. You can follow this procedure

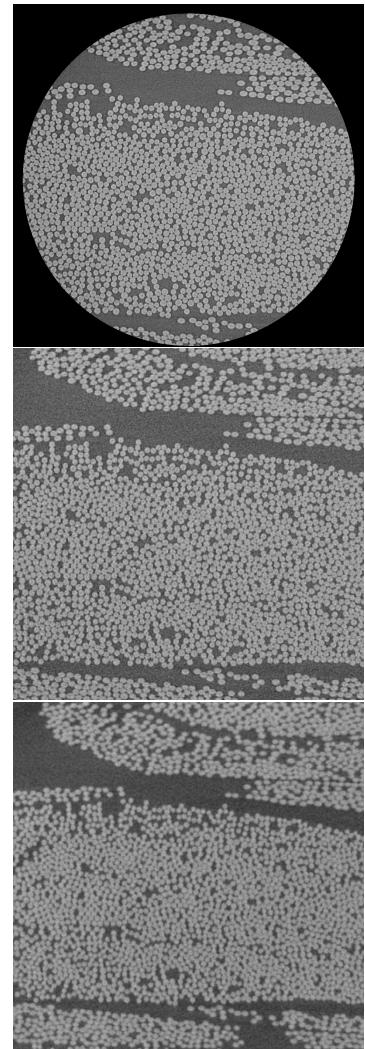


Figure 4.1: Example of an image of the same fiber sample acquired using a CT scanner at three resolutions.

K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (5):698–700, 1987

1. Generate a random 2D point set \mathbf{P} .
2. Define variables for translation \mathbf{t} , rotation \mathbf{R} , and scale s , and decide on their values.
3. Transform \mathbf{P} using these parameters to obtain the point set \mathbf{Q} .
4. Plot these point sets using two different colors.
5. Implement a function that computes the parameters \mathbf{t}' , rotation \mathbf{R}' , and scale s' from \mathbf{P} and \mathbf{Q} . Make sure that you get the exact same values.
6. Add some noise to \mathbf{Q} and recompute the parameters and test how much noise you can add and still get reasonably good parameter estimates.

4.1.2 Compute and match SIFT

Here you should compute SIFT features in two images and match them using Euclidean distance between their descriptor vectors. You should use the criterion by Lowe where a correct match is found, if the fraction between the closest feature vector and the second closest feature vector is less than a certain value, e.g. 0.6. There are functionality for matching features in both `vlFeat` and `OpenCV` that you can use. You can also implement your own matching function where you take e.g. scale and rotation into the matching criterion. You can follow this procedure

1. Create a transformed image by rotating, scaling, and cropping an image. We call the original `image1` and the transformed `image2`.
2. Compute SIFT features in the two images.
3. Match the SIFT features. You can use the functionality for matching SIFT from `vlFeat` for MATLAB and `OpenCV` for Python.
4. Display the match to see if the matching criterion is correct.
5. Extract the coordinates of the matching keypoints.
6. Use the function for computing the rotation, translation, and scale from before to transform the set of key points found in `image1` to the set of keypoints found in `image2`.
7. Display `image2` and plot the key points found in `image2` and the transformed key points from `image1`.
8. When you have confirmed that the points match, you can try to match the CT-images of fibers at the three resolutions shown in Figure 4.1. Visualize the matching feature points by drawing lines between them e.g. as shown in Figure 4.2. This allows you to visually evaluate the matching.

4.1.3 Transform the matched features

Now you should combine the function for computing the transformation parameters and the SIFT feature matching. The matching will not be perfect and you will most likely see some wrongly matched features. The larger the difference in appearance or scale of the image that is being matched, the more wrongly matched features can be expected. If the majority of the correspondences are correct then the least-squares fit will give a relatively good result.

Since we are computing the transformation by a least-squares fit, the outliers will affect the result to some extent. Outliers can however be removed relatively easily. You can compute the Euclidean distance between the two point sets after you have aligned them. There you will see that most of the distances are relatively small. And if you remove matching points with a distance larger than a certain threshold, you can repeat the computation of the transformation and obtain higher precision in the matching. You should implement a function that makes this two-step computation of the transformation and choose a good criterion for a threshold.

Illustrate your transformed feature points by plotting the two point sets on top of each other in the image e.g. as illustrated in Figure 4.3. You should be able to see a difference in the precision of the matching after removing the outliers.

4.1.4 Transform detected fibers (optional)

You have now established a correspondence between the fiber images, and you can now compare the detected fibers from the exercise on scale-space blob detection. You can do this by

1. Compute transformation between two fiber images.
2. Detect blobs in the two images
3. Transform blob parameters (location and size) from one image to the other.
4. Match blobs and compare their individual sizes.

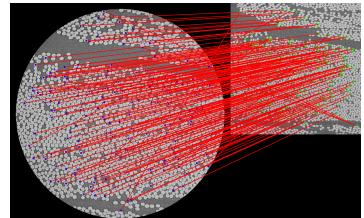


Figure 4.2: Matching SIFT features illustrated by red lines.

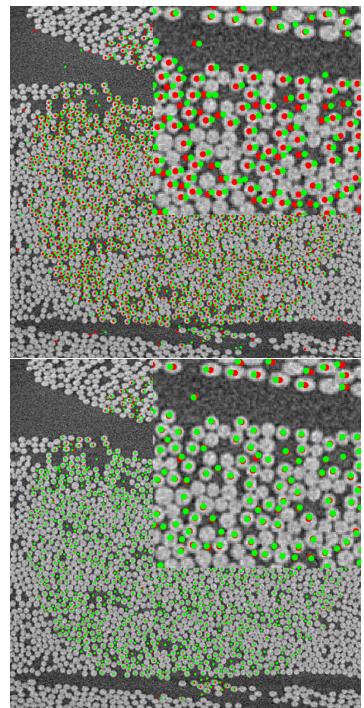


Figure 4.3: Matching features shown in red and green (zoom in upper right corner). Top is after computing least squares of all matching features and bottom is a recomputed match after removing outliers.