Scale-Invariant Feature Transform (Final Report)

Team 4: Drowning in Inefficiency

- Kannav Mehta (2019101044)
- Raj Maheshwari (2019101039)
- Triansh Sharma (2019101006)
- Aashwin Vaish (2019114014)

Github link. The repo contains the SIFT algorithm implemented in C++.

Project Objectives

The main objective of this project is to implement the SIFT algorithm described in the paper by David G. Lowe [1] from scratch (without any computer-vision dependencies). Distinct invariant features are extracted from images and matched with those from other views of the object or scene. These features are invariant to scaling, rotation, and give robust matching over a range of affine transforms.

SIFT

Following are the major stages of computation used to generate the set of image features using SIFT:

Scale-space extrema detection

Finding scale-invariant locations in the image can be done by searching for stable features across all possible scales. The scale space of an image is defined as a function, $L(x, y, \sigma)$, that is produced from the convolution of a an input image, $I(x, y)$ with a variable-scale Gaussian, $G(x, y, \sigma)$ using $*$ operator in x and y.

Gaussian function is the only possible scale-space kernel under reasonable assumptions as showed by Koenderink(1984) and Lindberg(1994).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Scale invariability is implemented to efficiently **detect stable keypoint locations** in scale space by computing $D(x, y, \sigma)$, which is obtained by convolving an image with a difference of Gaussian function to identify potential interest points that are invariant to scale and orientation.

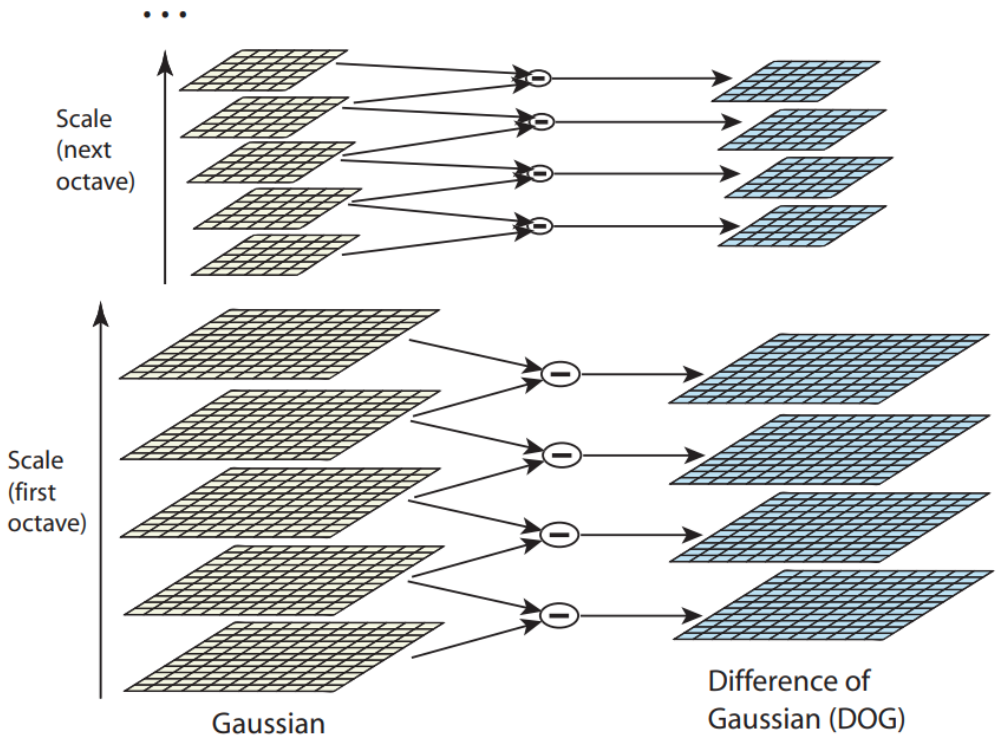
$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

Lindeberg(1994) showed that **normalisation of Laplacian** by σ^2 is required for **true scale invariance**.

The scale-normalized Laplacian $\sigma^2\nabla^2 G$, can be closely approximated to difference of Gaussian:

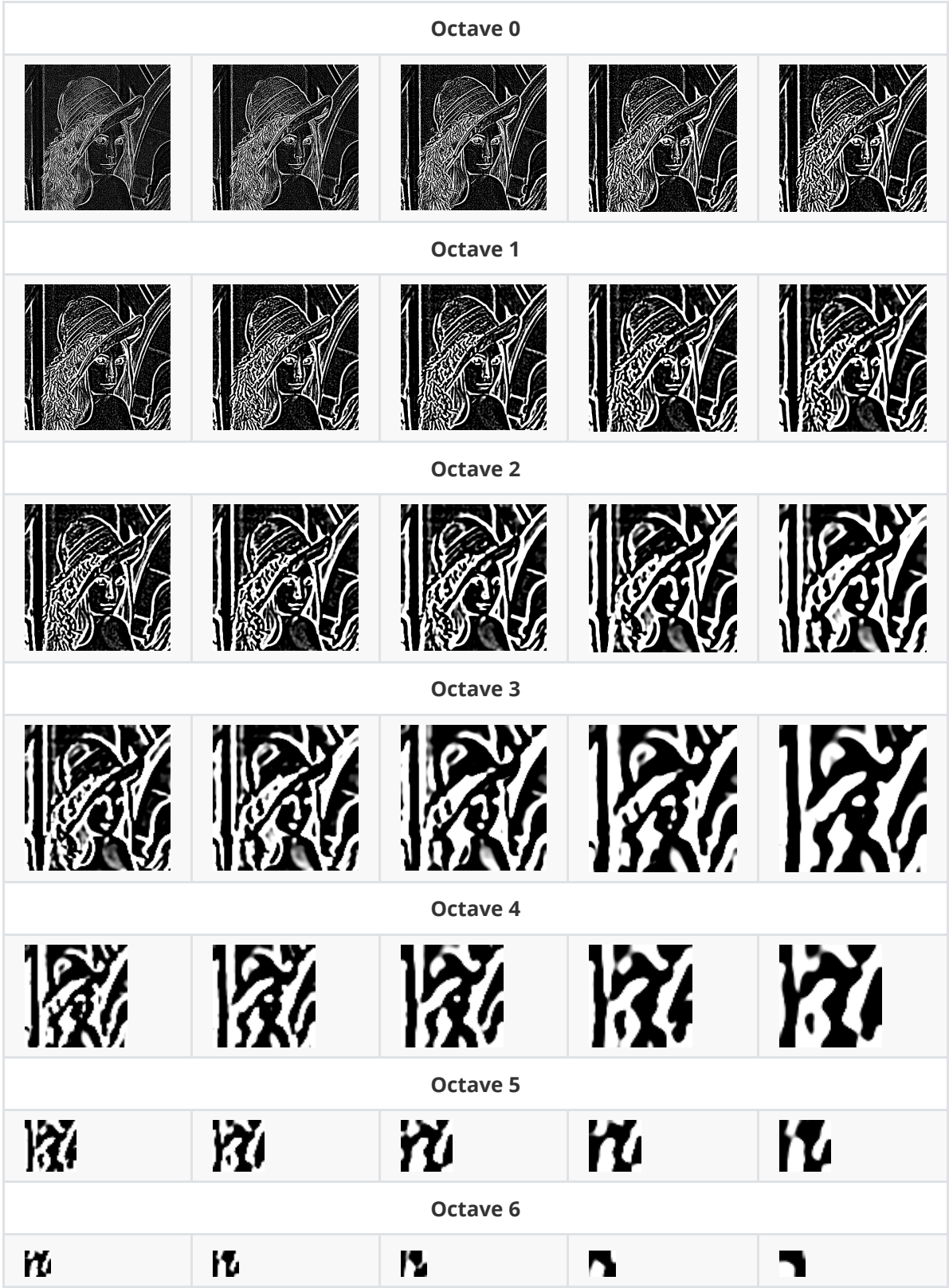
$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$
$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G$$

The constant factor $k-1$ over all scales does not influence extreme location. Theoritically, the approximation tends to equality when k tends to 1. However, practically there is no visible difference observed even with significant differences such as k=$\sqrt{2}$.

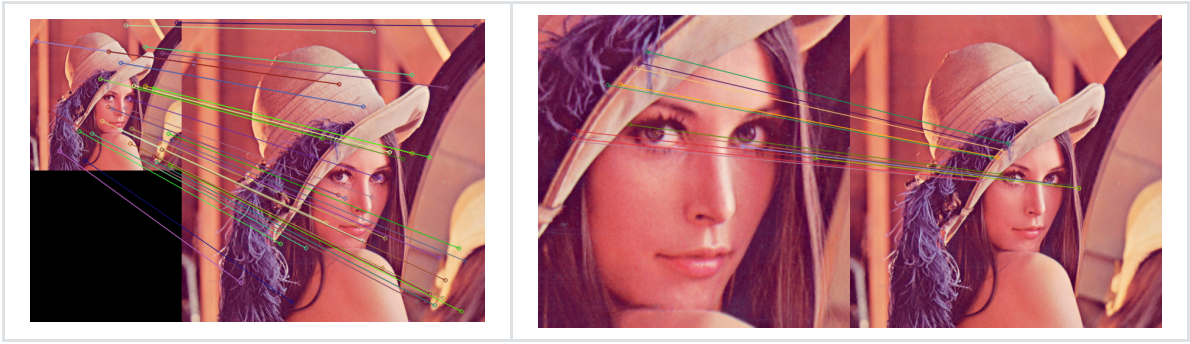


The images are incrementally convolved with Gaussian kernels as shown above. Every consecutive image differs by a factor of k.
The number of images in each octave should be such that the amount of blurring doubles between consecutive octaves. Let s be number of steps of blurring.
$k = 2^{\frac{1}{s}}$

DoG pyramid



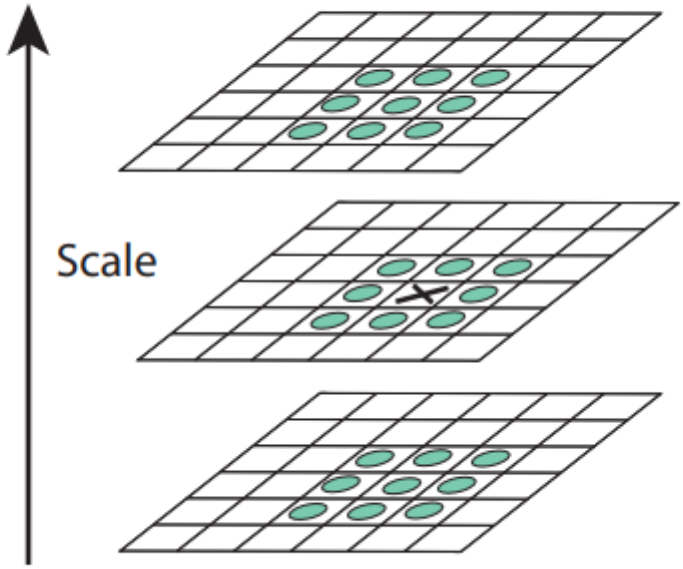
For s steps of blurring, we get $s + 1$ images. 2 additional images are required each for a step before the first and after the last images in the layer. From these $s + 3$ images, we compute the difference of Gaussians between adjacent images. The next octave starts with the scaled down to half size of the third last image.



Feature matching on upscaled and downscaled images (Scale Invariance)

Keypoint localization

Once each candidate is found by comparing its pixel value with its neighbours ($3 \times 3 \times 3$ cube centered on the pixel), a detailed model is fit to determine location, scale and principal curvatures.



Keypoints are selected based on measures of their stability. The location of the extremum, $\hat{x}$, is determined by taking the derivative of taylor expansion of previously obtained function $D(x, y, \sigma)$ with respect to x and setting it to zero, giving:

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}$$

where taylor expansion of $D(x, y, \sigma)$ is given by

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

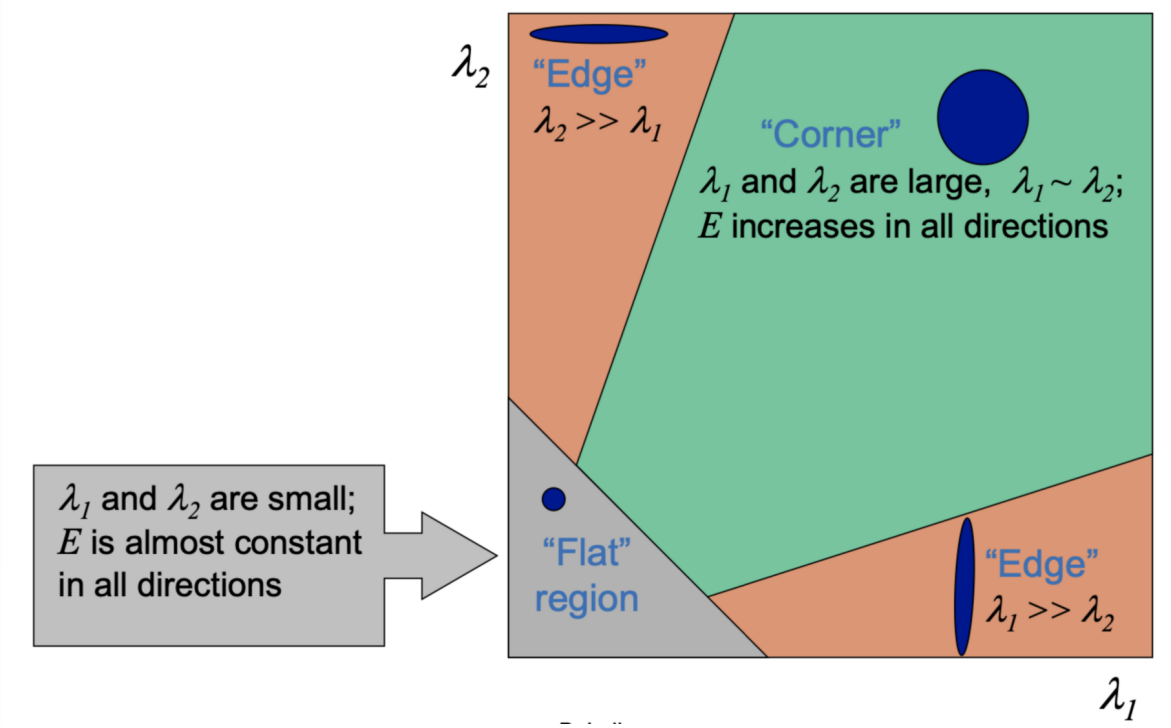
To obtain more stable fixed points, we apply a threshold on the value of $|D(\hat{x})|$ to reject key points with low contrast. Further, we eliminate those key points whose ratio for principal curvatures fall below a certain threshold.

We define Hessian H of D as

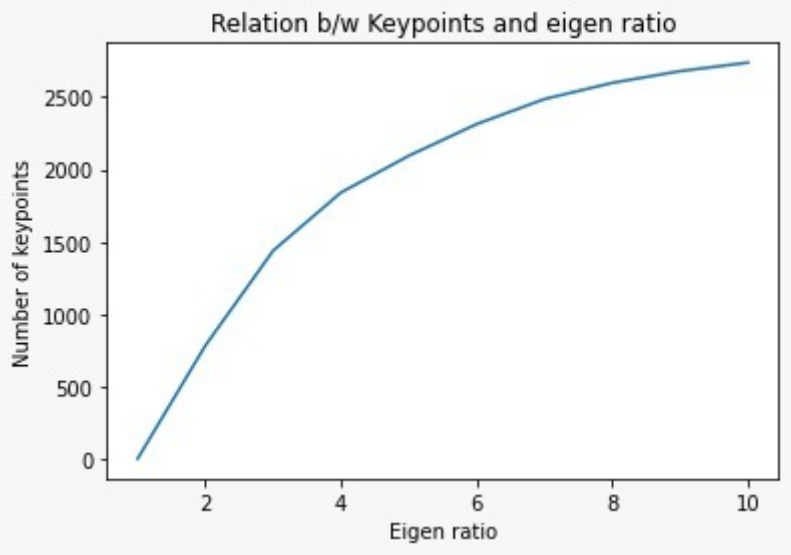
$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

The eigen values λ_1 and λ_2 of H are proportional to the principal curvatures of D, hence we define the threshold by τ, $(\lambda_1 = r\lambda_2)$

$$\tau = \frac{Tr(H)^2}{Det(H)} = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} < \frac{(r+1)^2}{r}$$



We modified the r from $1, 2, \cdots, 10$ and calculated the number of keypoints obtained for an image.
The below plot describes the nature of curve.



This shows that on increasing the threshold value, the number of keypoints increases and we obtain a curve which is similar to a parabolic graph.

Orientation assignment

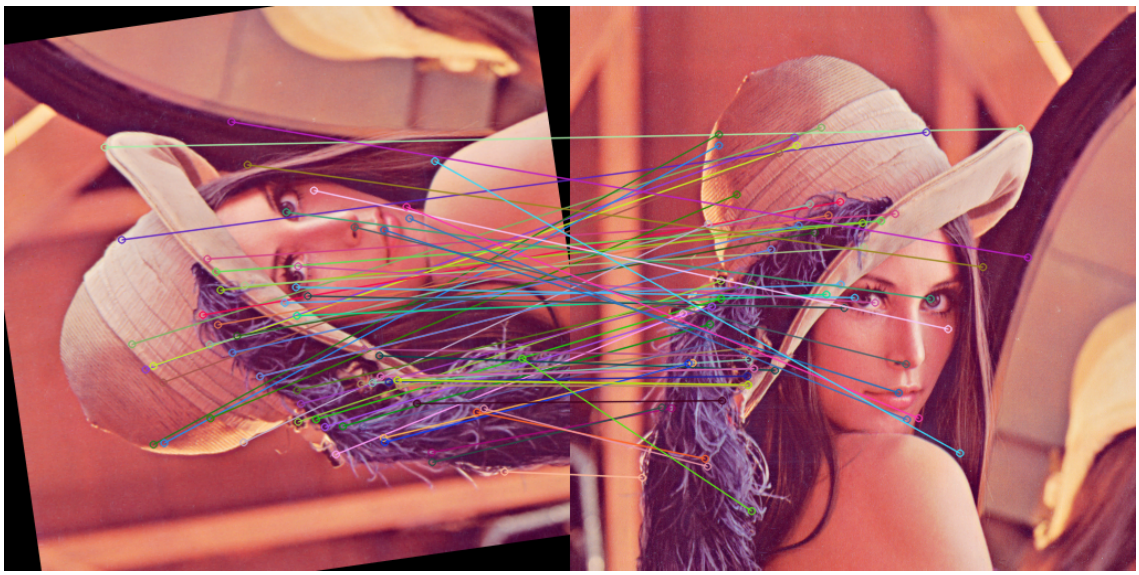
One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to each feature's set orientation, scale, and location, thereby providing invariance to these transformations.

The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, is precomputed using pixel
differences:

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

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

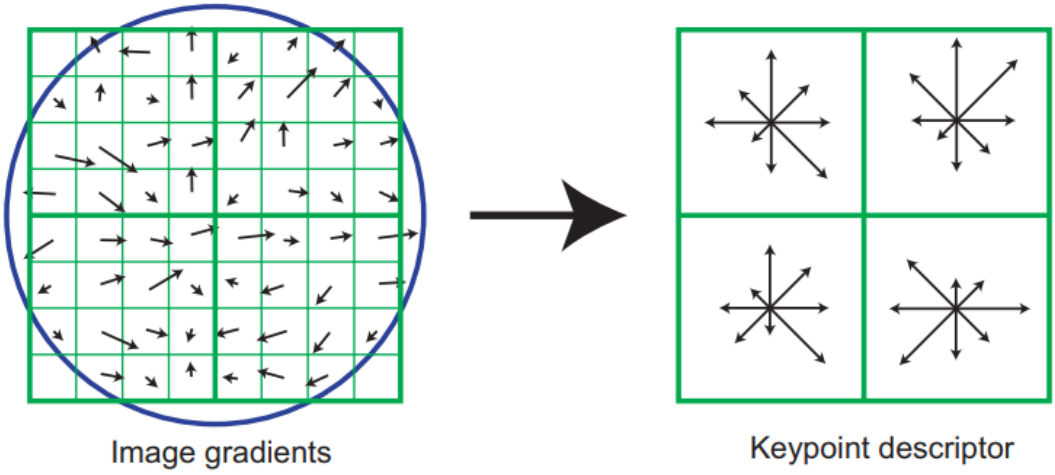
An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint. The orientation histogram has 36 bins covering the 360-degree range of orientations. We SHIFT the binned histogram by the orientation angle. That is, if the orientation angle is 93 degrees clockwise, we shift it $(93/10) \sim 9$ bins to the right. This provides **rotational invariance**.



Feature matching on rotated views (Rotational Invariance)

Keypoint descriptor

The local image gradients are measured at the selected scale in the region around each keypoint. These samples are then accumulated into orientation histograms summarising the contents over 4x4 subregions, as described in the figure, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region calculated using a Gaussian weighted function. This allows for significant levels of local shape distortion and change in illumination.



Feature Matching

Once features have been extracted from all n images (linear time), they must be matched. Since multiple images may overlap a single ray, each feature is matched to its k nearest neighbours in feature space. This can be done in $O(n\ log\ n)$ time by using a *k-d tree* to find the approximate nearest neighbours.
After the above step, we will have identified images that have a large number of matches between them and connected sets of these image matches that will later become panoramas.

Fast Approximate Nearest Neighbours (using FLANN)

The Fast Library for Approximate Nearest Neighbors (FLANN) is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It uses either one of two algorithms automatically depending on the situation, as they each perform predictably better depending on given situation. The two algorithm it selects from are :

- randomized kd-tree
- hierarchical k-means tree

Randomized kd-tree algorithm

The classic kd-tree algorithm consists of repeated binary partitioning the data on the dimension that shows the highest variance. However, the classic kd-tree algorithm is efficient for *low dimensions* only.
An improved but approximate version proposed by Silpa-Anan and Hartley (2008) creates multiple randomized trees from the top D dimensions with the highest variation. While traversal, a single priority queue is maintained across all the randomized trees so that search can be ordered by increasing distance to each bin boundary. The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned.

Hierarchical k-means tree algorithm

This algorithm works by recursively splitting the datapoints into K regions using k-means clustering. Recursion continues till the number of points in a region is less that K. An improved version proposed in Muja and Lowe (2009) explores the this tree in a best-bin-first manner. The algorithm initially performs a single traversal through the tree and adds to a priority queue all

unexplored branches in each node along the path. Next, it extracts from the priority queue the branch that has the closest center to the query point and it restarts the tree traversal from that branch. In each traversal the algorithm keeps adding to the priority queue the unexplored branches along the path. The degree of approximation is specified in the same way as for the randomized kd-trees, by stopping the search early after a predetermined number of leaf nodes (dataset points) have been examined.

Extra goals achieved

RootSIFT

In RootSIFT we use a square root (Hellinger) kernel instead of the standard Euclidean distance to measure the similarity between SIFT descriptors. RootSIFT is an enhanced SIFT descriptor.

Euclidean distance is a straight line distance between two pixels. The Hellinger distance between two points x and y is defined as:

$$H(x, y) = \sum_{i=1}^{n} \sqrt{x_i y_i}$$

Homography Estimation using RANSAC

RANSAC (random sample consensus) is a robust estimation procedure that uses a minimal set of randomly sampled correspondences to estimate image transformation parameters and finds a solution that has the best consensus with the data. In the case of panoramas, we select sets of r feature correspondences and compute the homography $\mathbf{H}$ between them using the direct linear transformation. Given the probability that a feature match is correct between a pair of matching images (the inlier probability) is p_i, the probability of finding the right transformation after n trials are:

$$P(\mathbf{H}\ is\ correct) = 1 - (1 - p_i^r)^n$$

After a large number of trials, the probability of finding the correct homography is very high. RANSAC is essentially a sampling approach to estimating $\mathbf{H}$. For each pair of potentially matching images, we have a set of geometrically consistent feature matches (RANSAC inliers) and many features inside the area of overlap but not consistent (RANSAC outliers). The idea of our verification model is to compare the probabilities that this set of inliers/outliers was generated by a correct image match or by a false image match.

SIFT features are extracted from all of the images. After matching all of the features using a k-d tree, the m images with the most significant number of feature matches to a given image are checked for an image match. First RANSAC is performed to compute the homography, and then a probabilistic model is invoked to verify the image match based on the number of inliers. For each pair that have a consistent set of feature matches, connected components of image matches are detected and stitched into panoramas.

Image Stitching

As a test for the verification of the Keypoints and the descriptors our implementation produces, we decided to use them to do image stitching. Firstly, the use of invariant features (SIFT features in this case) enables reliable matching of panoramic image sequences despite rotation, zoom and illumination change in the input images. Here are a few examples illustrating the use of the features that we generate to first do feature matching and then use the matches to find the homography and stitch the images.

Panorama with 2 images


Individual Images

Stitched Panaroma

Feature matching (168 good matches)

Panorama with 3 images



Individual Images

Stitched Panaroma

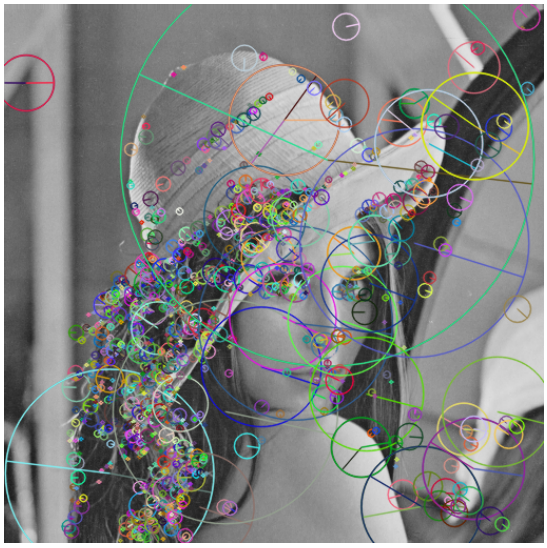
Observations

We analyzed multiple images by applying various filters and affine transformations and matched features to know the effectiveness of SIFT descriptors when matching using FLANN.

SIFT ensures rotational and scale invariance, whose examples are already mentioned above. Apart from it we judged the effectiveness of SIFT by modifying images on illumination, blurring and skewing. Following results are obtained.

Size: 512 × 512
Keypoints: 1112

Input image



Keypoints

Illumination Invariant

Feature matching under different brightness

Skew Invariant

SIFT descriptors are produced in such a way that they are able to obtain robust image matching in presence of affine transformations. Skew is such an example.

Feature matching under after skew transformation

Noise Invariant

Feature matching under after add artifical noise to the image. The noise was added to all the three channels.

Blurring Variant

Number of keypoints matched are reduced to 8. These correspond to the largest value maxima and minima in the scale space as smaller value maxima and minima have been eroded due to blurring.

Feature matching under different blurring

Work Distribution

1. **Kannav:** Gaussian Pyramid, DoG generation, Multithreading, Scale Space Extrema Detection, Sub-pixel Localization, Orientation Assignment, Keypoint Pruning, Descriptor Generation, Image Stitching & Feature Matching.

2. **Triansh:** Gaussian Pyramid, DoG generation, Sub-pixel Localization, Scale Space Extrema Detection, Orientation Assignment, Keypoint Pruning, Descriptor Generation, Image Stitching & Feature Matching.
3. **Raj:** Gaussian Pyramid, DoG generation, Scale Space Extrema Detection, Sub-pixel Localization, Orientation Assignment,Keypoint Pruning, Descriptor Generation.
4. **Ashwin:** Hessian, Gradient computation, Pixel Cube , Refactoring, Debugging.

References

1. *Brown, M., Lowe., D.G. Recognising panoramas. In proceedings of the 9th International Conference on Computer Vision (ICCV03), volume 2, pages 1218–1225, Nice, October 2003.*
2. *Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110 (2004).*
3. *Brown, M., Lowe, D.G. Automatic Panoramic Image Stitching using Invariant Features. Int J Comput Vision 74, 59–73 (2007).*
4. *Muja, Marius & Lowe, David. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration.. VISAPP 2009 - Proceedings of the 4th International Conference on Computer Vision Theory and Applications. 1. 331-340.*