The Intro to Computer Vision labs will be run in Google Colaboratory, a Jupyter notebook environment that runs entirely in the cloud, you don't need to download anything. To run these labs, you must have a Google account.

Step 1: click on the assignment invite link -> **Accept this assignment**. Refresh page -> individual repo for the specific assignment is created automatically

Project 2: 2 weeks
https://classroom.github.com/a/1ErT08UZ

Step 2: Navigate to http://colab.research.google.com/github -> Click the **Include Private Repos** checkbox -> **select the correct repo** (SistemeDeVedereArtificiala/assignment_name-student_name) -> Click on the jupyter notebook of the current assignment

Step 3: [GitHub sign-in window] In the popup window, sign-in to your Github account and authorize Colab to read the private files.

Step 4: [in colab] **File** -> **Save a copy to GitHub**. Select the correct repository for the SPECIFIC assignment -> Click the **Include Colab Link** -> Click **OK**

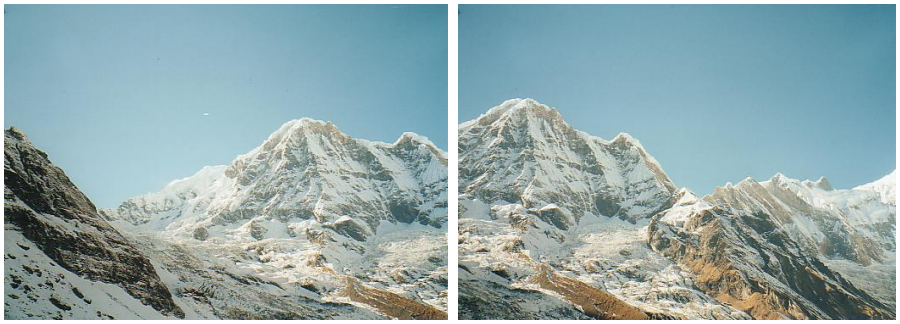
Step 5: [in colab] Navigate to the **Runtime** tab --> **Change runtime type**, under **Hardware accelerator** select **GPU/TPU** (tensor processing unit) according to your needs.

Read the suggestions and accomplish all tasks marked with **#TODO**.

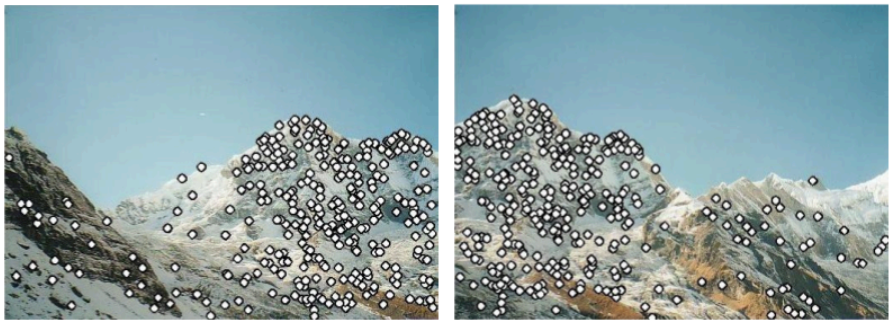
!!! At the end of each laboratory **REPEAT step 4 in order to SAVE** the answers to your private repository (individual for each assignment)

Project 3: Image Stitching

Q: We have two images – how do we combine them?

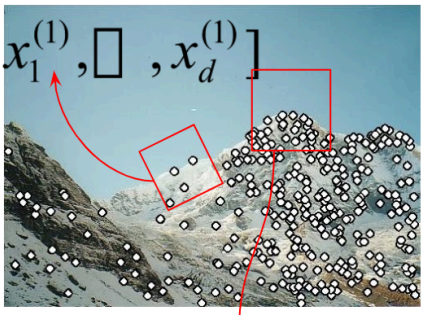


A: Image stitching using Local Features:



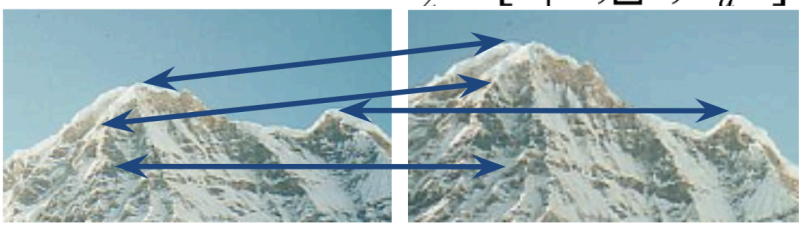
1. **Detection**: Identify the interest points
2. **Description**: Extract vector feature descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \square, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \square, x_d^{(2)}]$$

3. **Matching**: Determine correspondence between descriptors in two views



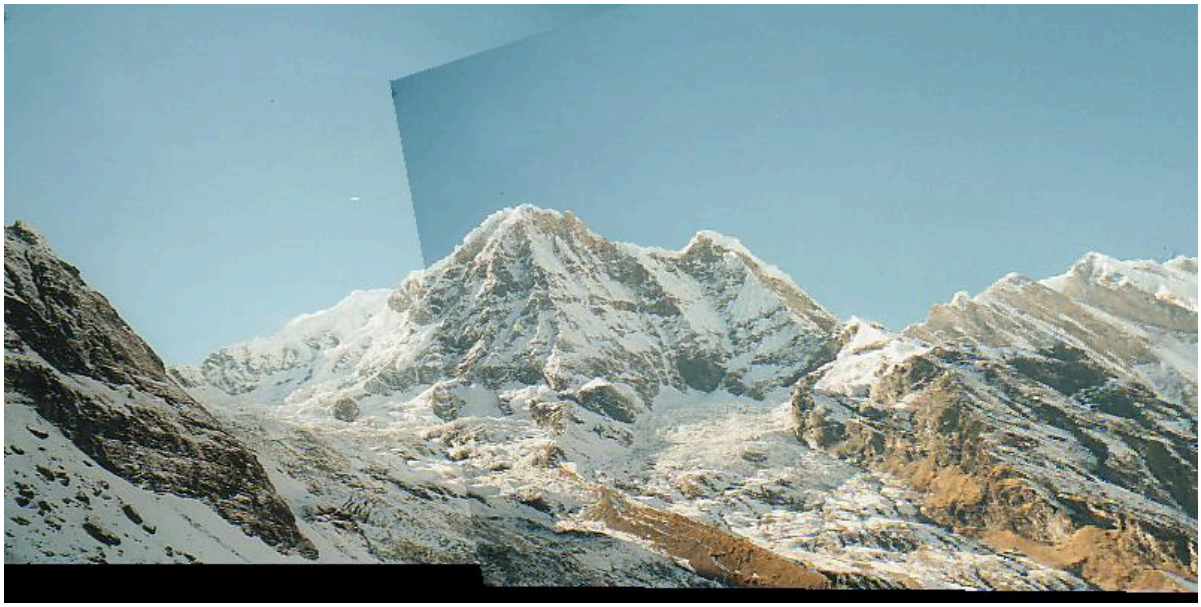


Figure: Stitching result - panorama

Week 1: Scale Invariant Feature Transform (SIFT) Algorithm

In 2004, **D.Lowe**, University of British Columbia, came up with a new algorithm, Scale Invariant Feature Transform (SIFT) in his paper, **Distinctive Image Features from Scale-Invariant Keypoints**, which extract keypoints and compute its descriptors. This paper is easy to understand and considered to be the best material available on SIFT.

In general, the SIFT algorithm can be decomposed into four steps:

1. Feature point (also called keypoint) detection
2. Feature point localization
3. Orientation assignment
4. Feature descriptor generation.

1. Feature point detection

SIFT has the property of scale invariance, which makes it better than Harris. Harris is not scale-invariant, a corner may become an edge if the scale changes, as shown in the following image.

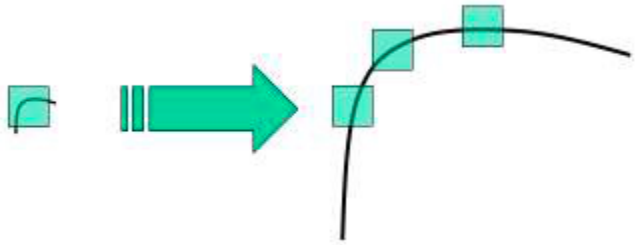


Figure: A corner can become an edge if scale changes

The scale of an image landmark is its (rough) diameter in the image. It is denoted by σ, which is measured in pixels, you can think of scale invariance as that we can detect similar landmarks even if their scale is different.

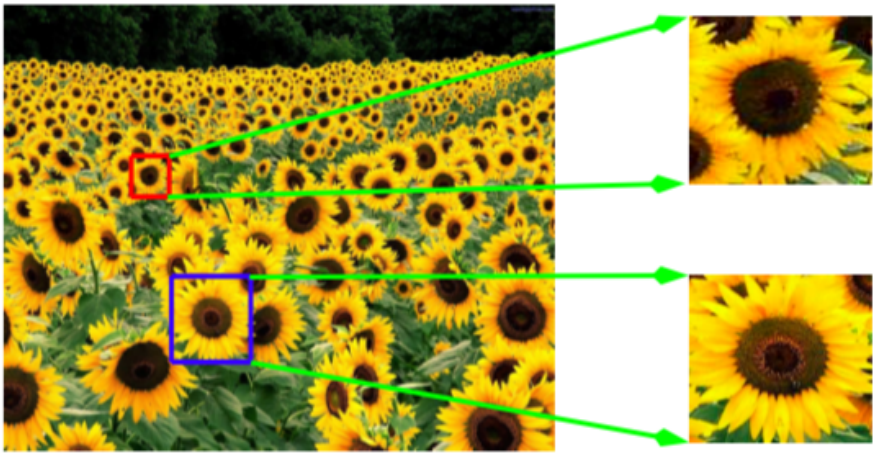


Figure: Similar features at different scales

Q: So how does SIFT achieves scale invariance?
A: We can find the features under various image sizes (see pyramid representation) and use the Laplacian of Gaussian (LoG) with different σ to achieve this.

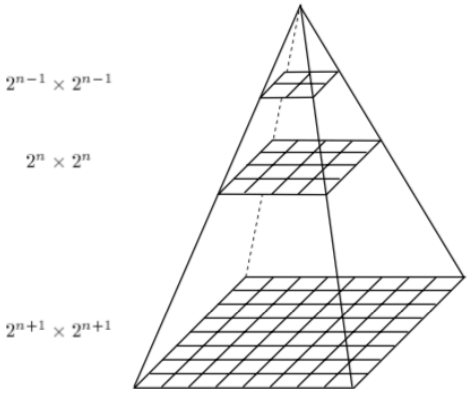


Figure: Pyramid representation is obtained by successively reducing the image size by combining smoothing and subsampling.

Figure: Matched SIFT descriptors on different scale images

2. Feature point localization (sub-pixel localization)

After step 1, we detect some key-points which are coarsely localized, at best to the nearest pixel, dependent upon where the features were found in the scale-space. They are also poorly localized in scale since σ is quantized into relatively few steps in the scale-space. The second stage in the SIFT algorithm refines the location of these feature points to sub-pixel accuracy whilst simultaneously removing any poor features. The sub-pixel localization proceeds by fitting a **Taylor expansion to fit a 3D quadratic surface** (in x,y, and σ) to the local area to interpolate the maxima or minima. Neglecting terms above the quadratic term, the expansion of the DoG (Difference of Gaussian) is given in following where the derivatives are evaluated at the proposed point z0=[x0,y0,σ0]T and z=[δx,δy,δσ]T is the offset from this point.

$$D(\mathbf{z}_0 + \mathbf{z}) \approx D(\mathbf{z}_0) + \left(\frac{\partial D}{\partial \mathbf{z}}\bigg|_{\mathbf{z}_0} \right)^T \mathbf{z} + \frac{1}{2}\mathbf{z}^T \left(\frac{\partial^2 D}{\partial \mathbf{z}^2}\bigg|_{\mathbf{z}_0} \right) \mathbf{z}$$

The location of the extremum zˆ is then determined by setting the derivative with respect to z equal to zeros:

$$\hat{\mathbf{z}} = -\left(\frac{\partial^2 D}{\partial \mathbf{z}^2}\Big|_{\mathbf{z}_0}\right)\left(\frac{\partial D}{\partial \mathbf{z}}\Big|_{\mathbf{z}_0}\right)$$

DoG has higher response for edges, so edges also need to be removed. For this, a concept similar to Harris corner detector is used. They used a 2x2 Hessian matrix (H) to compute the principal curvature. We know from Harris corner detector that for edges, one eigenvalue is larger than the other. So here they used a simple function.

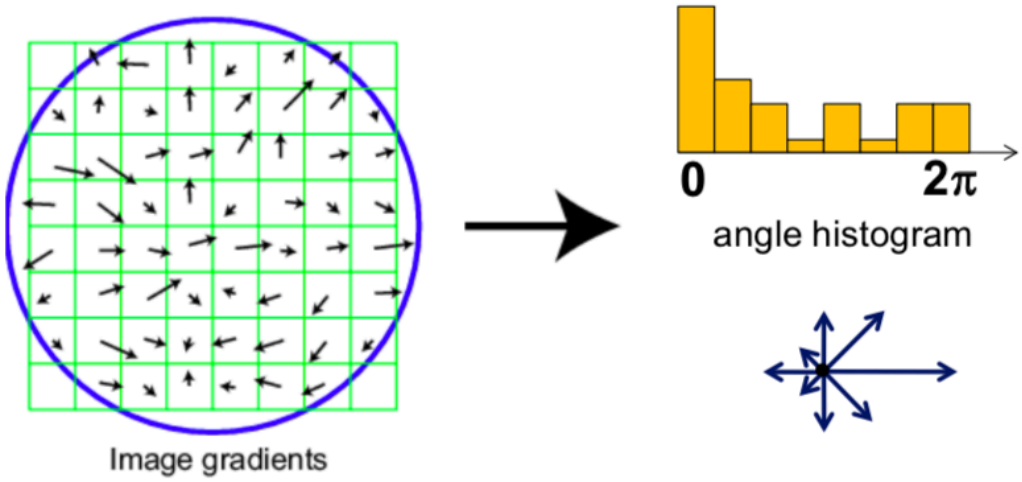
If this ratio is greater than a threshold, that keypoint is discarded. (threshold given as 10 in the paper)

So it eliminates any low-contrast keypoints and edge key points and what remains is strong interest points.

3. Orientation Assignment

We will use the Histogram of Oriented Gradient (HOG). 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. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with a σ that is 1.5 times that of the scale of the keypoint.

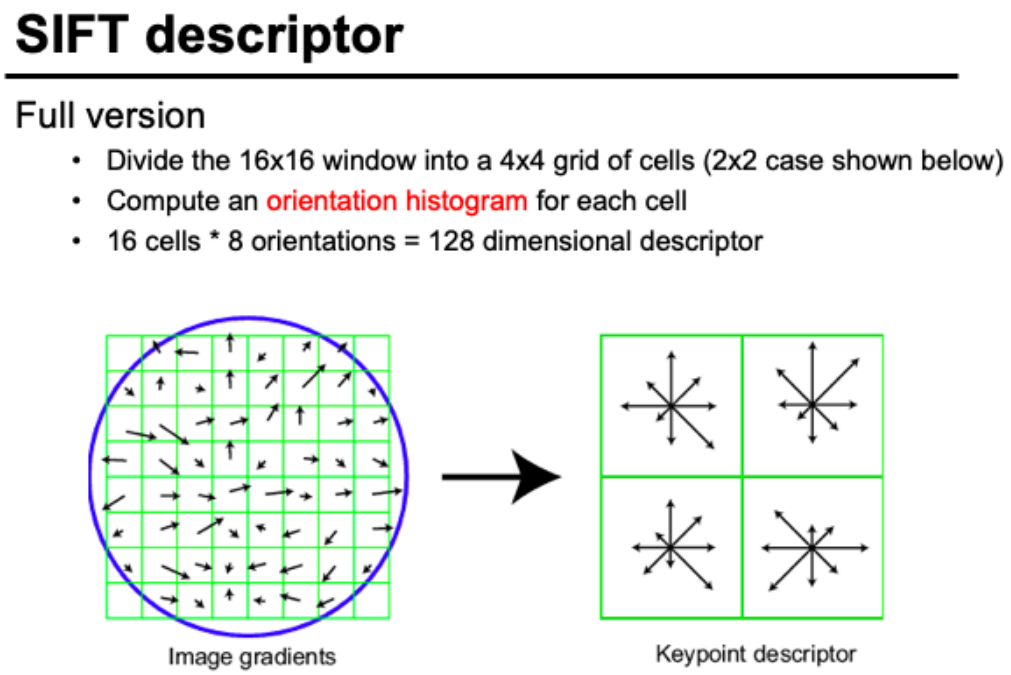
- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

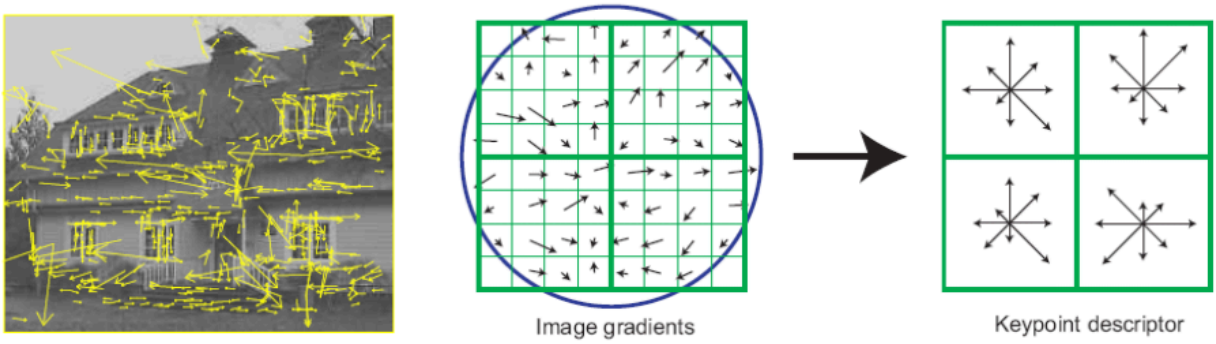


4. Feature descriptor generation

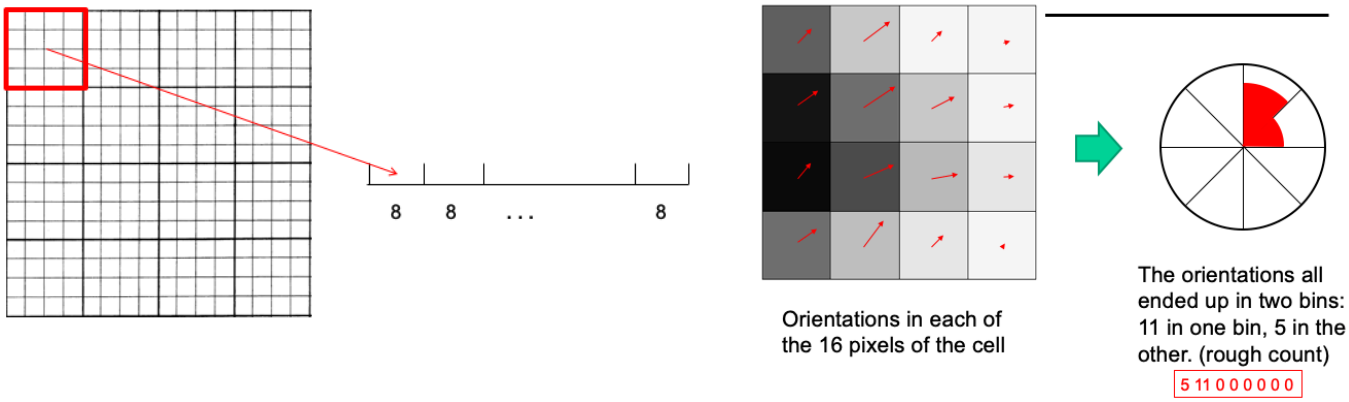
The final stage of the SIFT algorithm is to generate the descriptor which consists of a normalized 128-dimensional vector. At this stage of the algorithm, we are provided with a list of feature points which are described in terms of location, scale, and orientation. This allows us to construct a local coordinate system around the feature point which should be similar across different views of the same feature.

The descriptor itself is a histogram formed from the gradient of the grayscale image. A 4×4 spatial grid of gradient angle histograms is used. The dimensions of the grid are dependent on the feature point scale and the grid is centered on the feature point and rotated to the orientation determined for the keypoint. Each of the spatial bins contains an angle histogram divided into 8. (128=4×4×8). The image gradient magnitude and angle are again generated from the scale-space.





SIFT – Lowe IJCV 2004



Further study at:

1. Introduction to SIFT in OpenCV
2. Feature matching in OpenCV
3. Distinctive Image Features from Scale-Invariant Keypoints
4. Automatic Panoramic Image Stitching using Invariant Features