Lecture 8: Feature Descriptors and Resizing

Harrison Caruthers, Diego Celis, Claire Huang, Curtis Ogren, Junwon Park, Krithika Iyer
Department of Computer Science
Stanford University
Stanford, CA 94305
{hdcaruth, dcelis, chuang20, ceogren, junwonpk, ksiyer}@cs.stanford.edu

1 Scale invariant keypoint detection

1.1 Motivation

Thus far, we have covered the detection of keypoints in single images, but broader applications require such detections across similar images at vastly different scales. For example, we might want to search for pedestrians from the video feed of an autonomous vehicle without the prior knowledge of the pedestrians' sizes. Similarly, we might want to stitch a panorama using photos taken at different scales. In both cases, we need to independently detect the same keypoints at those different scales.

1.2 General methodology

Currently, we use windows (e.g., in Harris Corner Detection) to detect keypoints. Using identically sized windows will not enable the detection of the same keypoints across different-sized images (Figure 1). However, if the windows are appropriately scaled, the same content can be captured.

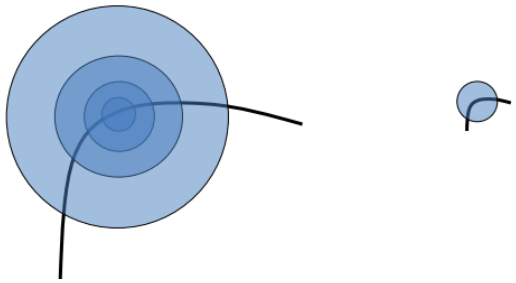

Figure 1: The corner of a curve appears at two scales. Note that the circular window on the right curve captures the entire corner, while the same-sized window on the left curve does not. Instead, we must choose a much larger circular window on the left curve to get the same information. Source: Lecture 7, slide 12.

How do we independently find the correctly scaled windows for each image? We need to describe what it means to "capture the same content" in a scale-invariant way. More specifically, consider a function, $f(\text{window})$, that takes in a region of content and outputs the same value for all scales of that region.

Now consider two similar images at different scales. We can independently vary window size for each of the images and plot the response of $f(\text{window})$ as a function of window size:

Within each of the two plots, we can independently identify local extrema as keypoints. The window sizes that correspond to those extrema (in the case of Figure 2, s_1 and s_2) provide us with the scale difference between the two images.

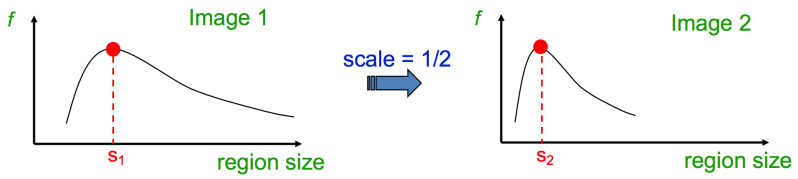

Figure 2: Two plots of the response of $f(\text{window})$ as a function of window size for Images 1 and 2, where Image 2 is similar to Image 1 but scaled by $\frac{1}{2}$. Source: Lecture 7, slide 15.

1.2.1 Average intensity

One candidate for such a function $f(\text{window})$ is the average intensity of the pixels within the window; this is because average intensity does not change as we scale the window up or down.

However, the average intensity is not great at capturing contrast or sharp changes within a window; this makes it harder to find clear extrema when comparing f across two images. To capture contrast, we need to bring in derivatives into the mix.

1.2.2 Difference of Gaussians

Another candidate would be to use the Difference of Gaussians method.

Consider an image I. First, the I is repeatedly convolved with Gaussian filters of different σ's. These convolutions are repeated with scaled down (i.e., down-sampled) versions of I. This results in the pyramid of Gaussians of different σ's and different image sizes (Figure 3). The adjacent Gaussian-convolved images are then subtracted to calculate their difference of Gaussians (DOG):

$$DOG(\sigma) = (G(k\sigma) - G(\sigma)) * I$$

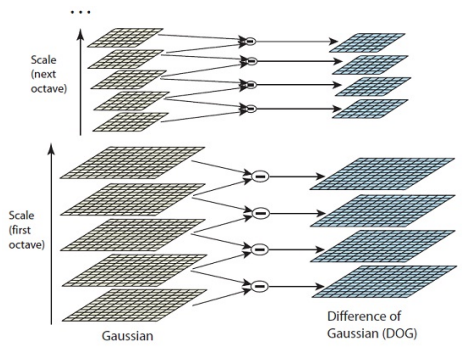

Figure 3: On the left: pyramid of Gaussians of different σ's and different image sizes. On the right: difference of adjacent Gaussians. Source: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-log-approximation/

Intuitively, these differences of Gaussians capture details about I at different scales. More specifically, the difference of two Gaussians σ_1 and σ_2 remove all details that appear at both σ_1 and σ_2 and keep only those details that appear between σ_1 and σ_2. The differences of Gaussians for small and large σ's respectively capture fine and coarse details.

Given the differences of Gaussians pyramid in x-y-scale space, we can now identify local extrema within that $3D$ space to identify both keypoints and their associated scales. We compare a given coordinate against its 26 neighbors (in $3D$ space) and deem it an extrema if it is smaller or larger than all neighbors (see Figure 4).

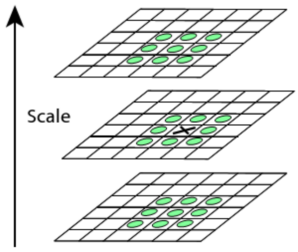

Figure 4: Given a coordinate in x-y-scale space (denoted by the black X), examine its 26 neighbors (denoted by the green circles) to determine if the original coordinate is a local extrema. Source: Lecture 7, slide 22

1.2.3 Harris-Laplacian

A third candidate is to use the Harris-Laplacian method [2], shown to be more effective at scale-invariant keypoint detection than the DoG but also potentially more computationally expensive.

Consider again an image I. First, we create multiple scales of I and run the Harris detector on each to localize keypoints per scale level. We then select the keypoints that maximize the Laplacian across all the scales.

Transition to scale-invariant descriptors: Now that we have several methods to *detect* consistent keypoints across multiple scales, we can move on to developing methods to *describe* those keypoints in a scale-invariant manner, so that they can be matched.

2 SIFT: an image region descriptor

2.1 Invariant Local Features

Point descriptor should be invariant and distinctive. To achieve robustness of point descriptors, we transform image content into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters .

The advantages of invariant local features include:

- **Locality**: features describe parts and are robust to occlusion and clutter (no prior segmentation).
- **Distinctiveness**: features are identifiable from a large database of objects.
- **Quantity**: many features can be generated for even small objects.
- **Efficiency**: close to real-time performance.
- **Extensibility**: can easily be extended to wide range of differing feature types, each adding robustness to changes.

2.1.1 Scale invariance

- The only reasonable scale-space kernel is a Gaussian. (Koenderink, 1984; Lindeberg, 1994)
- An efficient choice is to detect peaks in the difference of Gaussian pyramid (Burt & Adelson, 1983; Crowley & Parker, 1984 - but examining more scales)
- Difference-of-Gaussian with constant ratio of scales is a close approximation to Lindeberg's scale-normalized Laplacian (can be shown from the heat diffusion equation)

2.1.2 Rotation invariance

Given a keypoint and its scale from DoG,

1. Smooth (blur) the image associated with the keypoint's scale.

2. Calculate the image gradients over the keypoint neighborhood.
3. Rotate the gradient directions and locations by negative keypoint orientation. In other words, describe all features relative to the orientation.

2.1.3 SIFT descriptor formation

Using precise gradient locations is fragile, so we want to produce a similar descriptor while allowing for generalization. We create an array of orientation histograms and place gradients into local orientation histograms of 8 orientation bins. Dividing gradients into 8 bins is recommended, and this number of bins was found to exhibit best performance through experimentation.

More concretely,

1. Create an array of orientation histograms
2. Put rotated gradients into local orientation histograms, where each gradient contributes to the nearby histograms based on distance; gradients far from center are scaled down. The SIFT authors [3] found that the best results were achieved using 8 orientation bins per histogram and a 4x4 histogram array (Figure 2).
3. Compare each vector between two images to find the matching keypoints.
4. To add robustness to illumination changes in high contrast photos, normalize the vectors before the comparison. This mitigates the unreliable 3D illumination effects such as glare that are caused by the very large image gradients; this is achieved by clamping the values in vector to under 0.2 (an experimentally tuned value) before normalizing again.

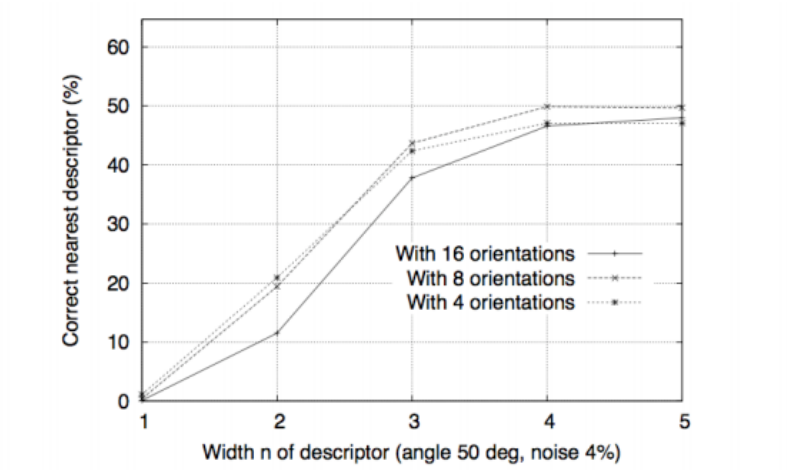

Figure 5: This figure shows the percentage of correctly matched keypoints as a function of the width of the descriptor and of the number of histogram bins. [1]

3 HoG: Another image region descriptor

3.1 Histogram of Oriented Gradients

The histogram of oriented gradients (HOG)[4] Descriptor finds an object within an image that pops-out, an object that can be discriminated. The general algorithm for HOG proceeds as follows:

1. Divide the image window into small spatial regions or cells.
2. For each cell, accumulate a local histogram; group gradient directions into evenly-spaced bins, and allocate the magnitude of a pixel's gradients into the appropriate bin corresponding to the gradient direction (Figure 6).
3. Normalize the local histograms over a larger region, called a "block", comprised of a number of cells.

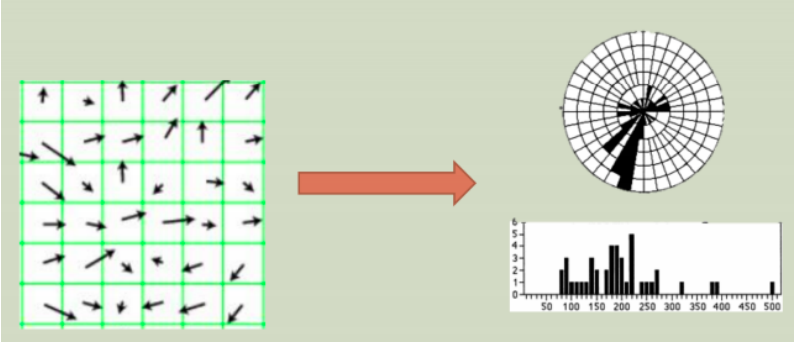

Figure 6: Here we see a visual example of keeping track of the magnitudes of the gradients for each gradient direction. Source: Lecture 7, Slide 60.

There are a few downsides to using HOG:

1. Large variations and ranges when detecting
2. Very slow
3. Not very organized when the backgrounds have different illuminations

Despite these downsides, HOG can be quite effective. Note the results of applying HOG in the image below.

Figure 7: HoG applied to a bicycle. Source: Lecture 7, Slide 65.

3.2 Difference between HoG and SIFT

There are some minor differences between the two. HOG is used over an entire image to find gradients. SIFT is used for key point matching. The SIFT histograms orient towards the natural positive gradient while HOG does not. HOG uses neighborhood bins, while SIFT uses weights to compute varying descriptors.

4 Image Stitching and Panorama Creation

4.1 Homogeneous Coordinates

Stitching together multiple images to create a panoramic view requires an understanding of projective geometry and homography. In addition to our familiar 3D Euclidean space, an additional dimension called W needs to be considered. W may be thought of as the distance between a projector and an image. The four dimensional space is called "projective space" and the coordinates in projective space are called "homogeneous coordinates". Conversion between image coordinates and homogeneous coordinates is carried out as follows: (i) to convert to homogeneous coordinates,

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

and (ii) from homogeneous coordinates to image coordinates

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (\frac{x}{w}, \frac{y}{w})$$

When taking pictures with a camera, W may be thought as the distance from the camera to the objects in the picture. Objects near the camera appear larger (in the image) that objects that are farther from the camera. This phenomenon is known as "perspective". Images from different "perspective"s need to be transformed to a common perspective before they can be stitched together.

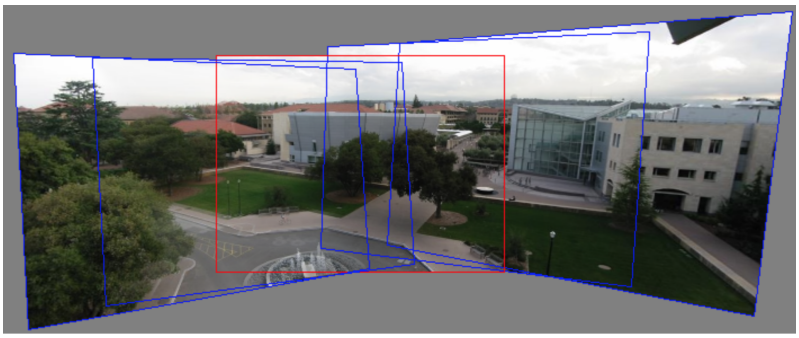

Figure 8: The different rectangles show images taken from different perspectives, but transformed to a common perspective plane through a keypoint matching and image transformation process. Source: $http://graphics.cs.cmu.edu/courses/15-463/2010fall/$

4.2 Transformations

The transformation of an image from one projective plane to another may involve translation, scaling (up or down), rotation, shear, and changes in aspect ratio. Such operations are illustrated below.

The matrices (in homogeneous coordinates) used to calculate such transformations from one perspective to another are shown below.

4.3 Homography

Homography matrix describes the transformation between points in one picture and a related neighboring picture (different perspective). In Figure 11, the identical points are denoted by their respective perspective based coordinates. Homography matrix H describes the transformation from one set of coordinates to the other.

The coordinates of the keypoint denoted by the red dot in both the images are related by the Homography matrix H. The point $p_1 = [x_1, y_1]$ on the left image (in Figure 11) matches with the point $p_2 = [x_1', y_1']$ on the right image, the transformation from one to the other is given by:

$$\begin{bmatrix} wx' \\ wy' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

It may be impossible to to find the transformation matrix H that transforms every point in image 2 to the corresponding point in another image. It is possible to estimate the transformation matrix H with least squares. Given N matched keypoint pairs, X_1 and X_2 (both $NX3$ matrices whose rows are homogeneous coordinates), H can be estimated by solving the least squares problem:

$$X_2 H = X_1$$

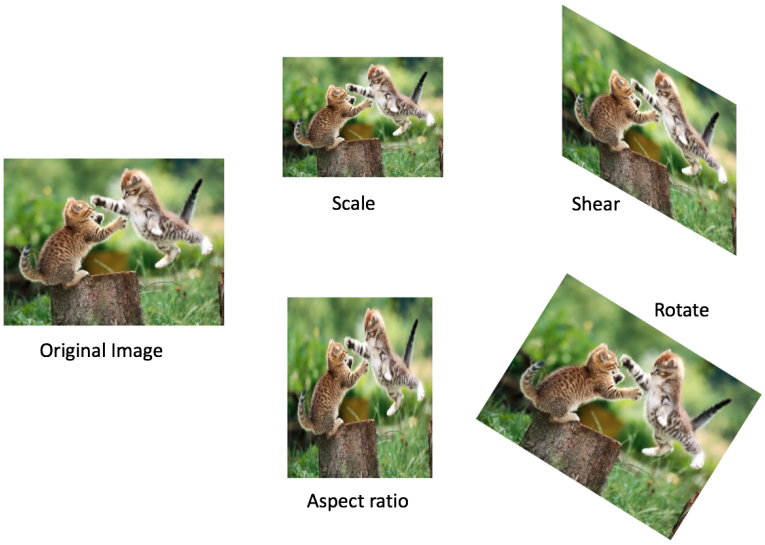

Figure 9: Basic transformation operations. Effect of 2D translation (which moves the image in x, y directions) is not shown here. Image Credit: S. Seitz and R. Collins. Penn State.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_x & 0 & 0 \\ 0 & \boldsymbol{s}_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

$$\begin{bmatrix} \boldsymbol{x}' \\ \boldsymbol{y}' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{sh}_x & 0 \\ \boldsymbol{sh}_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \\ 1 \end{bmatrix}$$

Shear

Figure 10: Matrices used to compute the new coordinates after basic transformations. Image Credit: Svetlana Lazebnik. UIUC.

4.4 RANSAC

During the least squares based estimation of the H matrix, it is a good practice to utilize RANSAC ("RANdom SAmple Consensus") to reduce the effect of outliers in the data. We avoid the impact of outliers by looking for inliers. The reasoning is that when an outlier is chosen to estimate the current fit, there will not be much support from the inliers for the current fit. Inliers are defined as points that are less than a threshold distance t, from the current fit.

The RANSAC steps are: (Source: Homework # 3)

1. Select a random set of matches.
2. Compute affine transform matrix.
3. Find the inliers using given threshold.
4. Repeat and find the largest set of inliers.
5. Re-compute least square estimates on all inliers.

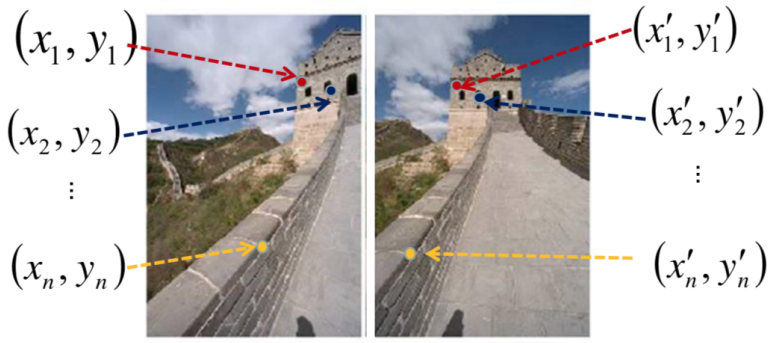

Figure 11: Identical points in different images and perspectives. Image Credit: Svetlana Lazebnik. UIUC.

5 Image resizing with seam carving

Because there are different screen sizes, we need to resize content according to display capabilities. Normally, we try to force content to fill up any kind of display by stretching or shrinking an image. However, this produces less than desirable results. So what is the solution? Content-aware re-targeting operators.

Retargeting means that we take an input and "re-target" to a different shape or size. Imagine input as being an image of size $n \times m$ and the desired output as an image of size $n' \times m'$. The idea behind retargeting is to

1. adhere to geometric constraints (e.g., aspect ratio),
2. preserve important content and structures, and
3. limit artifacts.

However, what is considered "important" is very subjective, for what may be important to one observer may not be important to another.

5.1 Pixel energy

A way to decide what is considered "important" is using **saliency measures**. There are many different types of saliency measures, but the concept is the same: each pixel p has a certain amount of "energy" that can be represented by the function $E(p)$.

The concept is that pixels with higher energy values are more salient, or more important, than pixels with lower energy values. What actually goes into the heart of E is up to the beholder.

A good example is to use the gradient magnitude of pixel p to heavily influence $E(p)$, for this usually indicates an edge. Since humans are particularly receptive to edges, this is a part of the image that is potentially valuable and interesting, compared to something that has a low gradient magnitude. As a result, this preserves strong contours and is overall simple enough to produce nice results. This example of E for image I could be represented as

$$E(\mathbf{I}) = \left|\frac{\partial \mathbf{I}}{\partial x}\right| + \left|\frac{\partial \mathbf{I}}{\partial y}\right|.$$

5.2 Seam carving

Let's assume that we have an input image with resolution $m \times n$ and we are looking for an output image $m \times n'$, where $n' < n$. How do we know what pixels to delete? We can use this concept of pixel energy to identify paths of adjacent pixels, or **seams**, that have the lowest combined pixel energy to remove from the image.

Note that the seams need not be strictly rows and columns. In fact, most of the time seams are curves that go through an image horizontally or vertically. A seam is horizontal if it reaches from the bottom edge to the top edge of an image. Similarly, a seam is vertical if it reaches from the left edge to the right edge of an image. However, seams are always laid out in a way such that there is only one pixel per row if the seam is vertical, or only one pixel per column if the seam is horizontal.

In essence, a seam avoids all important parts of an image when choosing what to remove from the image so as to cause the least disruption to the image when removed. There are more things to consider regarding seam carving and use cases that can be improved with similar techniques, but this is the core idea of how seams operate.

References

[1] David G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

[2] Mikolajczyk, Krystian. "Detection of Local Features Invariant to Affine Transformations." Perception Group, Institut National Polytechnique de Grenoble, INRIA Grenoble Rhône-Alpes, 2002.

[3] Lowe, David G. "Object recognition from local scale-invariant features." Computer vision, 1999. The proceedings of the seventh IEEE international conference on. Vol. 2. Ieee, 1999.

[4] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.