#5703: Computer Vision Homework 2

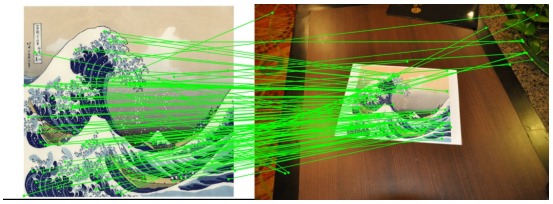
Due Date: Mar. 5th, 2022

SIFT, RANSAC, and augmented reality. Image: "The Great Wave off Kanagawa" by Hokusai.

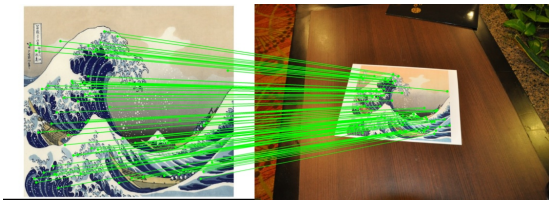
(a) Object template image.

(b) Test image.

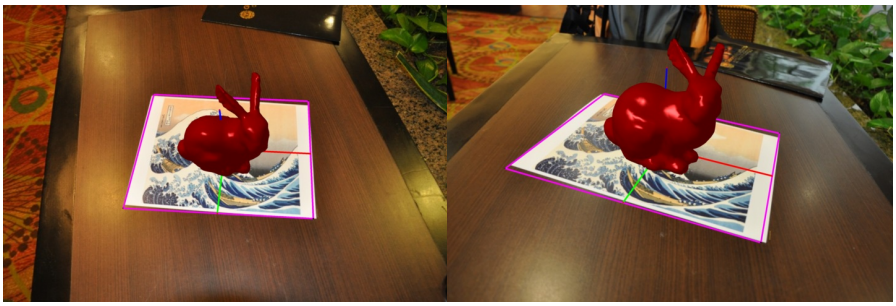
(c) SIFT keypoint matches.



(d) Homography constrained matches ("inliers").



(e) Recovered 3D pose and augmented views with the Stanford bunny.



Interest point detectors and descriptors are at the heart of many of the more successful applications of computer vision. SIFT in particular has been a boon to vision research since its appearance in Lowe's paper [2].

The goal of this assignment is to explore the use of interest points in several applications. We will be revisiting some of the camera geometry we used way-back-when, and apply it now to do augmented reality and object pose estimation. Along the process, we will try to make some pretty pictures that you can keep in your wallet and show off at parties and such.

This homework is adapted from Martial Hebert

1 Harris corners

A simple (and yet very useful[1]) interest point detector is the Harris corner detector. Implementing this is quite straightforward and we'd hate for you to miss out on the opportunity:

For every pixel, we compute the Harris matrix

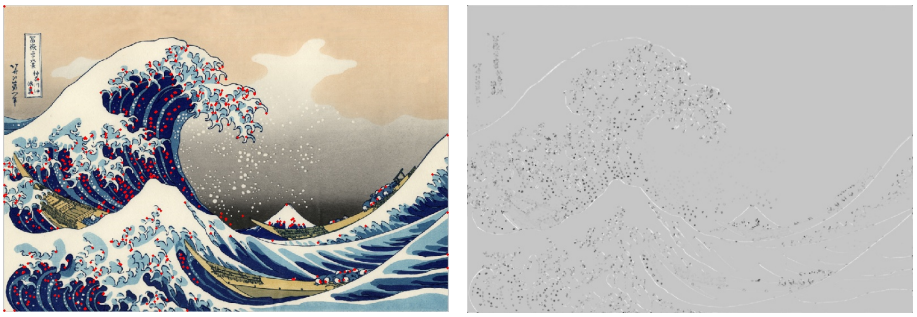
$$\mathbf{H} = \begin{pmatrix} \sum \mathbf{I}_x \mathbf{I}_x & \sum \mathbf{I}_x \mathbf{I}_y \\ \sum \mathbf{I}_x \mathbf{I}_y & \sum \mathbf{I}_y \mathbf{I}_y \end{pmatrix}, \tag{1}$$

where the sums are computed over a window in a neighborhood surrounding the pixel, and will be weighted with a Gaussian window. Remember that this matrix will be calculated efficiently using convolutions over the whole image. The "cornerness" of each pixel is then evaluated simply as $\det(\mathbf{H}) - k\,(\mathrm{trace}(\mathbf{H}))^2$ (easily vectorized).

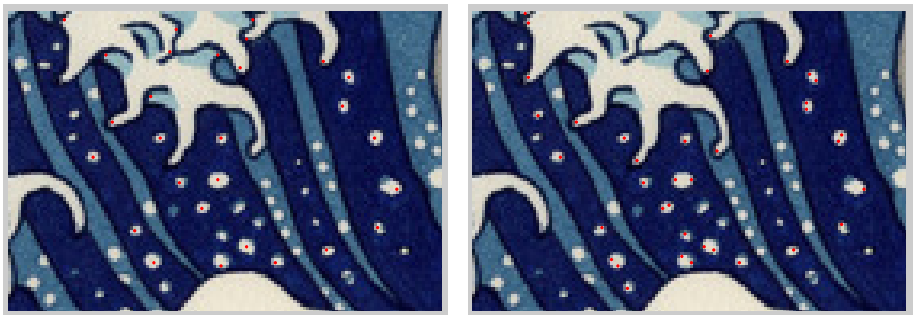
Refer to your notes—the algorithm is described in full detail. I also encourage you to revisit the derivation, it's really quite beautiful how it all works out in the end.

Figure 1: Harris corners. The importance of scale is apparent in the bottom close-up views. (Zoom in to see.)

(a) First 500 detected corners (red dots). (b) "Cornerness" heatmap (darker is cornier).



(c) Close-up view, one set of parameters. (d) Close-up view, a different set.



Q1.1 Discuss the effect of the parameters σ_x (the Gaussian sigma parameter used when calculating image gradients), $\sigma_{\sigma'}$, (the sigma for th neighborhood window around

[1] A *corner*stone method, one could say. *groan*

each pixel), and k.

Q1.2 Implement your own brand of Harris corner detection. (You can choose the parameters as you wish, or refer to the paper [1], with $k = 0.04$ a typical value (empirical)). Follow the skeleton given in `harris.m` and `p1_script.m`. Save the resulting images as `q1_2_heatmap.jpg` and `q1_2_corners.jpg`.

2 SIFT keypoints and descriptor

Because you'd have no time to explore the applications if we ask you to implement SIFT, we have provided you with Andrea Vedaldi's implementation of SIFT interest point detection, and SIFT descriptor computation. This function and many others can be found in Vedaldi's (and contributors) excellent `VLFeat` library [3]. I encourage you to browse the full package, it has implementations for many computer vision algorithms, many at the cutting edge of research (for this assignment though, we'll ask that you only use the functions provided).

I also suggest—if you're interested—that you study Lowe's paper in detail, especially two concepts that you won't easily find outside of computer vision: (1) the scale-space, and (2) using histograms of gradient orientations as a descriptor of a field.

The function is used like this (after doing `addpath ./sift`):

```
[keypoints1,descriptors1] = sift( double(rgb2gray(im1))/255, 'Verbosity', 1 ) ;
```

Verbosity is not necessary but the function takes a while to compute. (You can save the results after running once for each image.) See `sift_demo.m`

3 Image matching, revisited.

This is similar to the approach we followed in homework. Instead of using a bank of filter responses, we will now use the SIFT descriptor as our feature vector, and instead of computing features per image pixel, we will only compute the features at the scale-invariant interest points[2].

This is going to be a very crude object detector—an *objet d'art* detector, in fact. For each training image (in `objets/`), we will compute SIFT points and descriptors. Then, for each test image (in `images_test/`), we will do the same.Because our dataset is so small, we will directly compare all the test SIFT descriptors with all the training descriptors for each training image. We will label the testing image as the image with most.

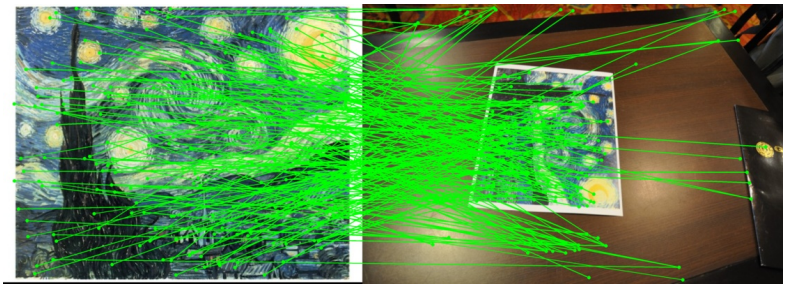
Q3.1 Write the function `[matches, dists]=matchsift(D1, D2,th)` that matches the set of descriptors `D1` to `D2` (follow the skeleton file given). Use pdist2 to compute distances. *matches* is $2 \times K$ where, $matches(i, k)$ is the index in the keypoint array of image i, of the k-th match or correspondence, for a total of K matches.

[2]Sometimes SIFT descriptors are computed densely for every pixel, similarly to homework 2.

To make matching a bit more robust, we will implement a commonly heuristic: a descriptor i in `D1` will be said to match j in `D2` if the ratio of the distance to the closest descriptor (call it $dist(i,j)$) and second-closest descriptor in `D2` (call it $dist2(i,j)$) is smaller than α (i.e., $\frac{dist(i,j)}{dist2(i,j)} \leq \alpha$). A common value is $\alpha = 0.8$. Note: Vedaldi provides a similar `siftmatch` function, you can use it to test but you are expected to implement your own.

Q3.2 For the 6 test images (in `images_test/`), display the matches side by side with the training image (in `objets/`) with most matches, as below. (Use the function plotmatches(im1, im2, keypoints1, keypoints2, matches), where *matches* is $2 \times N$ as above.). Save these images as `q3_2_match#.jpg`, substituting the testing image number.

Figure 2: SIFT keypoint matches. Image: "Starry night", from Van Gogh.



References

[1] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[2] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[3] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.