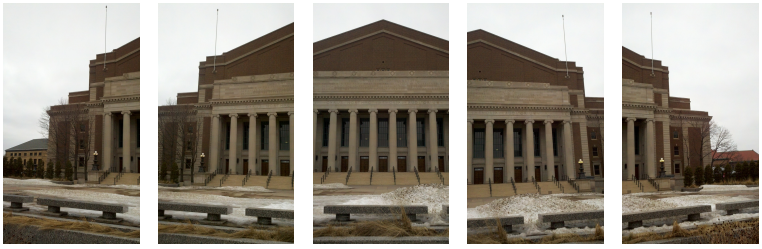
**UNIVERSITY OF TEXAS AT ARLINGTON**
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**6367** **COMPUTER VISION** **SPRING 2019**

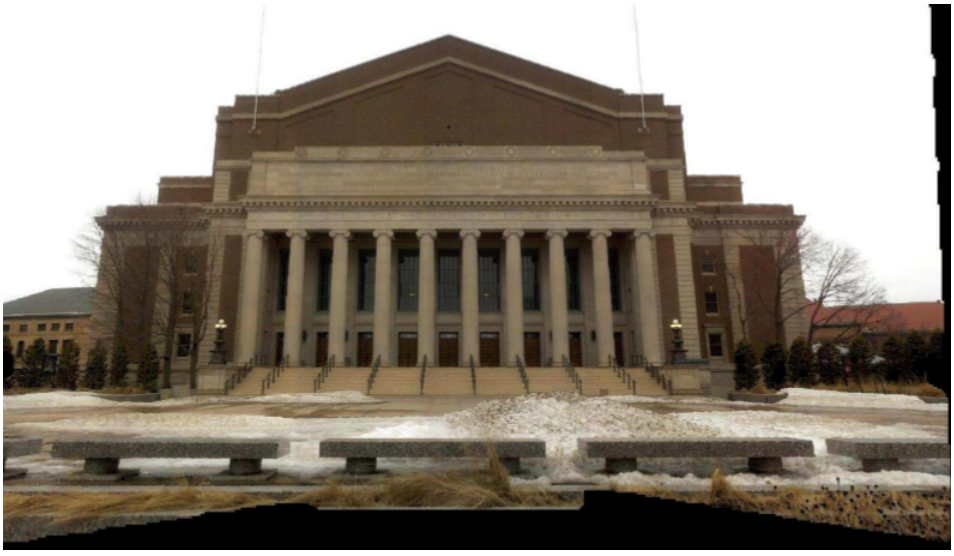
# ASSIGNMENT 3 (100 POINTS)
# ASSIGNED: 3/19/2019 DUE: 4/2/2019

This assignment constitutes 10% of the course grade. You must work on it individually and are required to submit a PDF report along with the MATLAB scripts described below. Your programs must not use computer vision functions provided by MATLAB unless otherwise specified.

In this assignment, you will write an image panorama program. Given a set of input images (Figure 1 (a)-(e)), your program must first extract the features, match the features across images, and compute the homography between them using these features. After applying the homography, you must then blend them together so that the image boundaries appear smooth and produce a single composite image (Figure 1 (f)). To do this, follow the procedures outlined below.



(a) mall1 (b) mall2 (c) mall3 (d) mall4 (e) mall5



(f) panorama

Figure 1: (f) is the composite image obtained by stitching together images (a)-(e).

# 1 Feature Detection and Matching

**Problem 1.1 Corner Detection (10 points)**

Write a MATLAB function `[pts] = extract_corners(img,THRESH)` that returns the locations of all corner points in an image, `img`, using the Harris corner detection algorithm. Use `THRESH` as the threshold on the response value of the detector. `pts` is a vector containing the locations of all corner points determined using non-maximal suppression. You may use the MATLAB function `imgradientxy` to compute the image derivatives.

**Problem 1.2 Corner Matching (10 points)**

Write a MATLAB function `[mpts1,mpts2] = match_corners(img1,img2,pts1,pts2,WSIZE)` that takes as input two images, `img1` and `img2`, along with their detected corner locations, `pts1` and `pts2` (found using the `extract_corners` function), and a window size, `WSIZE`. The output of the function will be two vectors, `mpts1` and `mpts2`, containing the location of corresponding corners in the two images. Specifically, `mpts1` and `mpts2` contain the location of the corresponding corners in `img1` and `img2` found by computing the normalized cross correlation. The normalized cross correlation is defined as

$$NCC = \frac{(f - \bar{f})(g - \bar{g})}{\sqrt{\sum (f - \bar{f})^2 \sum (g - \bar{g})^2}},$$

where $f$ is a corner point in the first image, $g$ is a corner point in the second image, and $\bar{f}$ and $\bar{g}$ are the mean intensity values within a window of size `WSIZE` centered at $f$ and $g$ in their respective images. The NCC is computed for every pair of corner points where the larger the value of the correlation the more likely that the corner points are matches.

**Problem 1.3 SIFT Features (10 points)**

Use the `vl_sift` function from the SIFT toolbox available at www.vlfeat.org to detect the SIFT features in an input image. Match the obtained SIFT features from two images using the `vl_ubcmatch` function. In the PDF report, submit two images marking the location of the detected features and the matches found. You can show the matches using arrows in the two images placed side by side, or mark the position of the features using text labels on the image. Use the input images 'mall1.jpg' and 'mall2.jpg' for finding the features.

# 2 Homography Estimation

Using the matching points `mpts1` and `mpts2` found in the previous sub-problems, we can compute the homography between the input images. In general, the correspondences found may contain a number of outlier matches. In the presence of outliers, the least squares or RANSAC method can be used to determine the homography.

**Problem 2.1 Least Squares Method (15 points)**

Write a MATLAB function `H = compute_homography(mpts1, mpts2)` that takes as input two $N \times 3$ vectors of homogeneous image coordinates, and returns as output the $3 \times 3$ homography matrix such that `mpts2 = H mpts1` using least squares fitting. Note that the number of input points, $N$, must be greater than or equal to four in order to estimate the homography.

**Problem 2.2 RANSAC Method (15 points)**

Write a MATLAB function `H = compute_homography_ransac(mpts1,mpts2,THRESH)` that uses the RANSAC algorithm to reject the outlier matches in `mpts1` and `mpts2`, and then uses only the inliers to find $H$. Since $H$ can be computed using four points, at each iteration the of algorithm you must sample four matches and

use these corresponding pairs to determine the candidate $\hat{H}$. Use the symmetrical reprojection error as the distance function to determine the inliers. The symmetrical reprojection error is defined as

$$d = ||mpts2 - \hat{H}mpts1|| + ||mpts1 - \hat{H}^{-1}mpts2||.$$

Since `mpts1` and `mpts2` are homogeneous coordinates, you have to normalize $\hat{H}$`mpts1` and $\hat{H}^{-1}$`mpts2` before computing the norm. To decide if a point is an inlier, use `THRESH` as a threshold for the error. If `THRESH` is 0, then use $2.5\sigma$ as the threshold where $\sigma$ is the standard deviation of the error. Try out different values of `THRESH` to see which ones give reasonable performance.

Use only the inliers found to compute the return value using the least squares `compute_homography` function. Additional information can be found in [1]. In particular, refer to steps 1-3 of Algorithm 4.6 in section 4.8 'Automatic computation of a homography'.

# 3 Image Blending

**Problem 3.1 Laplacian Pyramid Blending (30 points)**

After estimating the homography between two images, they can be stitched together using Laplacian pyramid blending. Write a MATLAB function `blend = blend_images(img1,img2,mask1,mask2)` that takes as input two images and their corresponding binary masks and blends them together into a single output image using five pyramid levels. For additional information, see section 3.5.5 in [2] (available online).

# 4 Image Panorama

**Problem 4.1 Composite Creation (10 points)**

To stitch together the left and right images related by a homography $H$, you must first transform the points in the right image to those in the left by applying $H$. Next, use the left and the transformed right images as input to the `blend_images` function. You may want to create larger images. Can you see what the mask images should be in this case?

Write a function `panorama = create_panorama2(img_left,img_right,use_sift,use_ransac)` using all of the functions created above that takes as input two images and finds the corresponding feature points, estimates the homography based on these points, and finally blends these images together to create a panoramic image as output. `use_sift` and `use_ransac` are boolean flags that determine whether to use corner detection or SIFT features and least squares or RANSAC methods.

Write a MATLAB script 'create_panorama.m' that reads in all five input images and uses `create_panorama2` repeatedly to create a single composite image automatically. To create visually better results you can add an extra input flag to the `create_panorama2` function that decides whether you transform points in the left image to the right image, or vice versa.

**Submission Instructions:** *Submit all of the MATLAB functions described above and any other MATLAB files necessary to run your program.* **Do not submit the SIFT toolbox or any image files**. *In addition, submit a PDF report containing the results of each of the operations separately along with the final stitched images. You may want to try your functions with other images and include these in the report.*

# References

[1] R. Hartley and A. Zisserman. "Multiple View Geometry in Computer Vision". Cambridge University Press, 2003.

[2] R. Szeliski. "Computer Vision: Algorithms and Applications". Springer Science & Business Media, 2010.