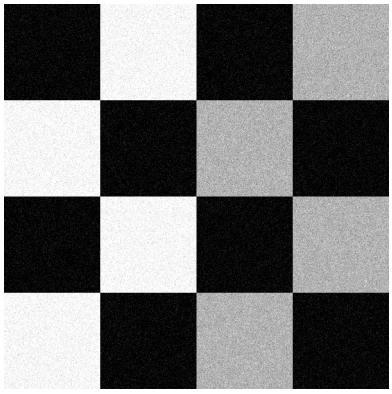
Project 4: Feature Detection and Matching

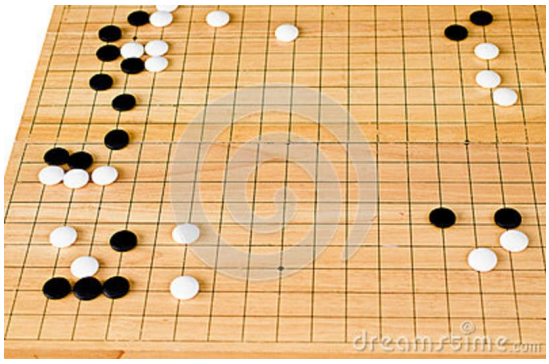
Each group should

- Submit one report per team containing procedures and results of the project by the end of Feb. 23.
- Include a thoughtful, reflective paragraph from each of the team members summarizing what has been learned.
- Include a section describing the work done by each member of the group.
- Attach your well-commented code to the report.

In this assignment, you will write code to detect keypoints in an image and find the best matching keypoints in the other image. It has three parts:

1. **Feature Detection**:
   Write a program to implement Harris corner detection. You can not use MATLAB built-in functions other than fspecial() and imfilter(). Test your program on the checkerboard image and Go board image below, and overlay the detected corners on the image using red squares. Describe your effort in tweaking the parameters (such as Gaussian filter size, threshold value for validating a corner).

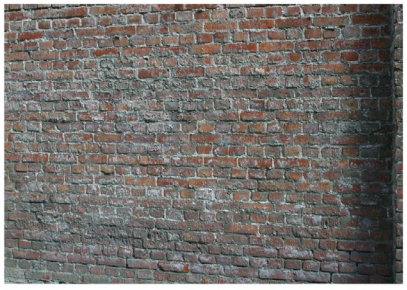
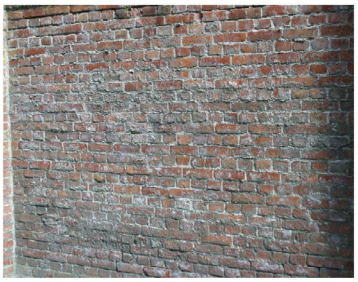




2. **Feature Description**:
   Write a function that extract the feature vector from the detected keypoints. The function should have the format: [extracted_features] = my_extractFeatures_a/b(image, detected_pts)
   For this part, you can use MATLAB's built-in feature detectors (such as FAST, SURF) to detect keypoints. Limit the number of detected keypoints to no more than 100 per image. Check the coordinates of the returned keypoints.
   a. First use raw pixel data in a small square window (say 5x5) around the keypoint as the feature descriptor. This should work well when the images you are comparing are related by only a translation.
   b. Next, implement a SIFT-like feature descriptor. You do not need to implement the full SIFT (for example no orientation normalization if no rotation is involved).





3. **Feature Matching**:
   Now that you have detected interest points and created the respective feature vectors, the next step is to match them in two images, i.e., given a feature in one image, find the best matching feature in the other image.
   a. Write a procedure that compares two features and outputs a distance between them.
   b. Find the best match for each detected keypoint in image 1.
   c. Your routine should return NULL or some other indicator if there is no good match in the other image. This can be done by
      i. a threshold on the nearest neighbor distance
      ii. the “ratio test”.
   d. Perform feature matching on the three pairs of test images shown above (files on Canvas). You can use showMatchFeatures() to display the matches.