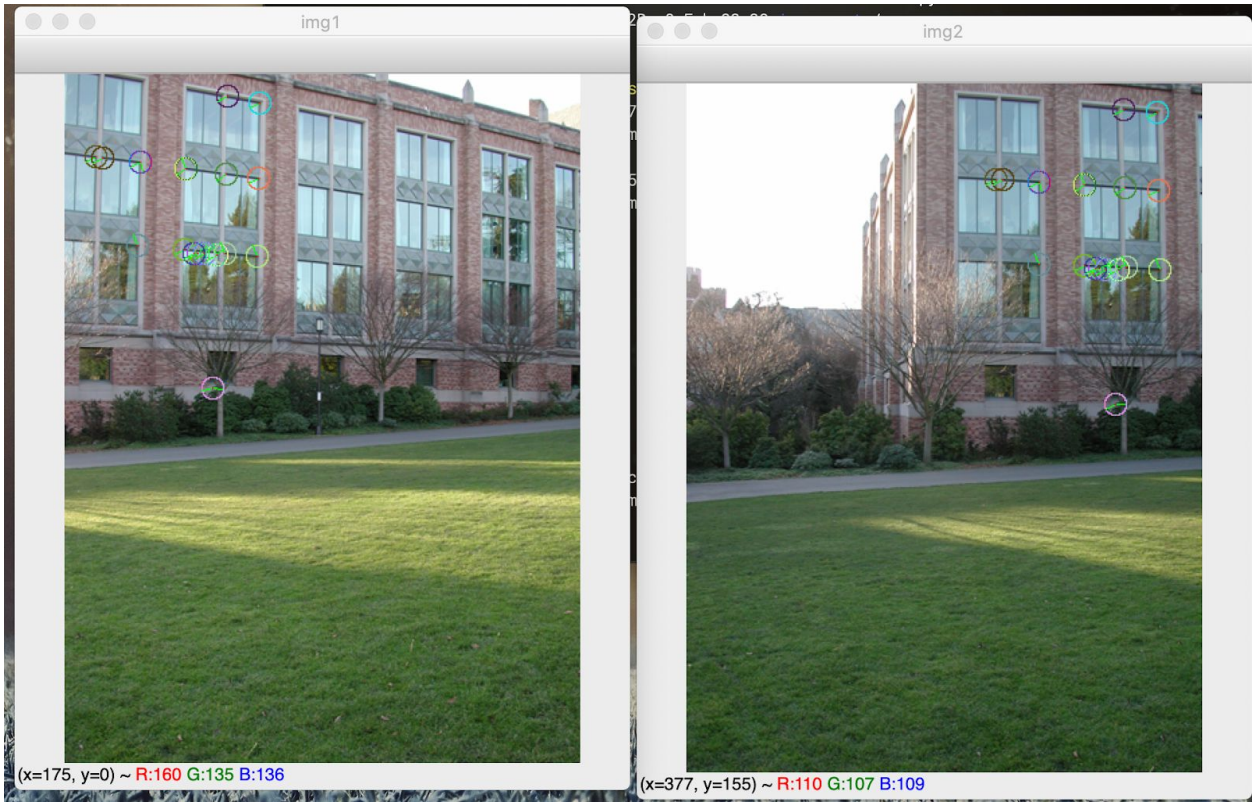
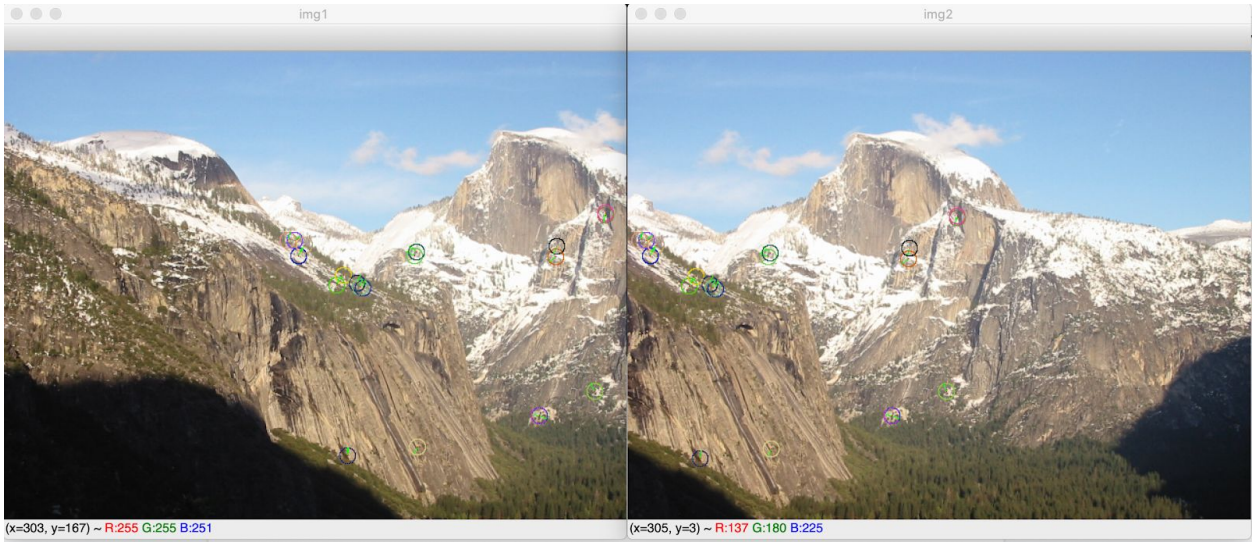
Program steps:

- ***Compute harris keypoints*** - uses 3x3 gaussian with sigma = 1, thresholds values that are lower than 20% of the max corner strength value, non-max suppression by checking against surrounding 3x3 grid, ignore points near (within 8 pixels) image boundary
- ***Compute orientation for keypoints*** - for rotation invariance, the dominant orientation for each keypoint is computed by histogram with 10 bins in a 10x10 window, weighted with a 10x10 gaussian of sigma = 3.5. If necessary, a single keypoint is split into multiple keypoints. Orientation value of the dominant bin is interpolated by fitting a parabola with the neighboring bins and getting the vertex.
- ***Compute descriptor for keypoints*** - first, gradient is subtracted from the orientation of the keypoint from previous step for rotational invariance. SIFT-like, 128 dimension vector descriptor is created by computing and concatenating orientation histogram in 4x4 sections of 16x16 neighborhood around the keypoint. Magnitudes are scaled by distance from the keypoint via 16x16 gaussian with sigma = 4. Each histogram is normalized and thresholded to mitigate the effect of large gradient values.
- ***Match keypoints*** - ‘distance’ between keypoints is computed by euclidean distance. The keypoint pairs with the smallest distance (threshold = 0.75) are selected and displayed on the screen with corresponding colors for visual confirmation / testing. Also, the ratio test is used to display the points for which the ratio between the distance of the best match and the second best match is sufficiently small (threshold = 0.9)

Results


img1
img2
(x=538, y=344) ~ R:220 G:222 B:223
(x=170, y=143) ~ R:121 G:49 B:36
cv.imshow("nonMaxSuppressed", nonMaxSuppressed)
cv.imshow("annotatedImage", annotatedImage)
master*
Python 3.8.0 64-bit


img1
img2
(x=175, y=0) ~ R:160 G:135 B:136
(x=377, y=155) ~ R:110 G:107 B:109


img1
img2
(x=303, y=167) ~ R:255 G:255 B:251
(x=305, y=3) ~ R:137 G:180 B:225