P2 Feature Detection, Feature Matching and Image Warping

Benjamin Geyer[1]

I. P2.1 Harris Corners

A. 2.1 Computing Harris Corners

Generate a figure (a 2x2 image grid) that shows the following intermediate steps during the computation of Harris Corners on the provided: 1. The original image 2. The x and y derivatives of the image, as computed via the Sobel filters. 3. The scoring function f 4. The original image with dots showing the location of the detected corners. (Be sure to choose a *reasonable* threshold; you should expect to end up with on the order of a few dozen features. If you have thousands of features, your threshold is too low. If you have only one or two features, your threshold is too high.

For this first part, you should use a *uniform weight matrix of size 5x5* (`weights = np.ones((5, 5))/25`) when computing the scoring function. We will change this in the next part of the question.

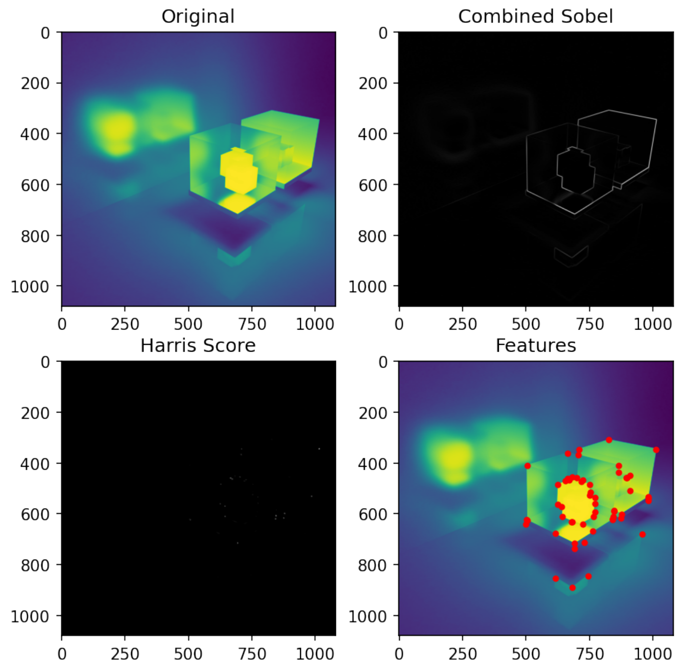

Fig. 1. 5x5 Uniform Harris Score Steps

QUESTION A: Are the features where you expected?

Yes, the algorithm detected most of the significant corners that I can see in the image.

QUESTION B (IMAGE REQUIRED): Are there any features in the image that you are surprised are not present? Highlight one or two regions of the image where features were detected that you did not expect (or one or two regions you thought features might exist).

I am surprised that the corners on the reflection on the bottom of the image are not picked up more. It got some of them but not others and they seem quite similar in terms or "cornerness" and brightness to me. I am also surprised that the corners in the smaller cube in the big cube on the right were detected since they are not very bright and are quite blurry.

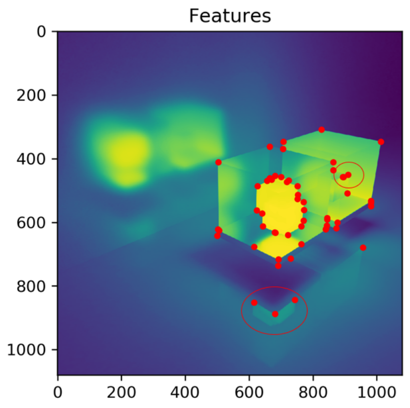

Fig. 2. Surprising 5x5 Uniform Harris Features

In class, we derived the Harris matrix and a special scoring function related to the eigenvalues of that matrix so that we could measure the *cornerness* of the image.

QUESTION C (IMAGES REQUIRED): What would happen if you used a scoring function $f = \text{tr}(H) = A + C$? Plot this alternative scoring function for the `light_cubes_base` image and plot the detected features computed using it.

Refer to Fig. 3. Alternative Harris Scoring Function

QUESTION D: What does this scoring function detect?

This scoring function detects both edges as well as corners. This is shown on the left in the score images where the edges are clearly detected in solid white along with the corners.

QUESTION E: If we want to detect corners, why might we not want to use this scoring function?

This alternate scoring function detects both edges and corners because it doesn't measure if A is much larger than C or vice versa in the H matrix. This means that if there is a large change in one direction but not the other such as on a straight edge, the scoring function will still give a large response. With a very high threshold, sometimes it only selects corners because the highest combination of change in both directions is usually a corner so the `get_local_maxima()`

will still select the corners. However, this alternative scoring function usually results in detecting many features along edges as well as corners.

B. P2.1.2 Varying the Weight Matrix

In this part, we will see what happens when we use different weight matrices. **Plot the score function f (one of the ones we discussed in class) and the detected corners for each of the following weight functions**:

1. A uniform weight matrix of size 5x5 `weights = np.ones((5, 5))/(5**2)` (same as in the previous question).
2. A uniform weight matrix of size 25x25 `weights = np.ones((25, 25))/(25**2)`
3. A Gaussian weight matrix with $\sigma = 5$
4. A Gaussian weight matrix with $\sigma = 50$

Refer to Fig. 3. Alternative Harris Scoring Function

QUESTION A: Discuss the differences between these four weight functions. In particular, what happens when the filter width (or σ) is very large?

The result of using Uniform instead of Gaussian is that there is more variance along an edge so the local maxima function detects many local maxima along the edge resulting in many detected corners.

Using a larger sigma allows for more noise to be blurred out resulting in less false corners. However, when using a sigma that is very large (>25 in this case), many corners other than the largest ones are lost since there is a smoother response leading to each maximum.

QUESTION B: What happens if we were to use a 1x1 weight matrix $w = 1$? Why does this occur?

The determinant of the H matrix becomes 0 in the scoring function and all of the information is lost. This happens because

$$I_x^2 * I_y^2 = (I_x I_y)^2$$

when only considering 1 pixel at a time. When considering more than 1 pixel to look at, the diagonals are no longer equal and the difference between them is no longer 0.

II. P2.2 Multi-scale Blob Detection

Here, we will be building on the in-class breakout session to experiment with using the (normalized) Laplacian of Gaussian (LoG) filter to detect "blobs" in an image and their scale.

A. P2.2.1 Scale-Normalized Filter Response

In this question, I have provided you with a simple "circle image", in which a filled circle is placed at the center of a square image. Here, you know that the circle is the feature you are trying to detect and that its location is at the center, so the feature does not need to be *located*. Instead, you are asked to **find the radius of the circle** (the "blob feature" of interest).

The steps are detailed below:

1. Once you have computed the normalized LoG filter, apply it at multiple scales to the sample circle image I have provided you with and **plot a few of the filtered images**.

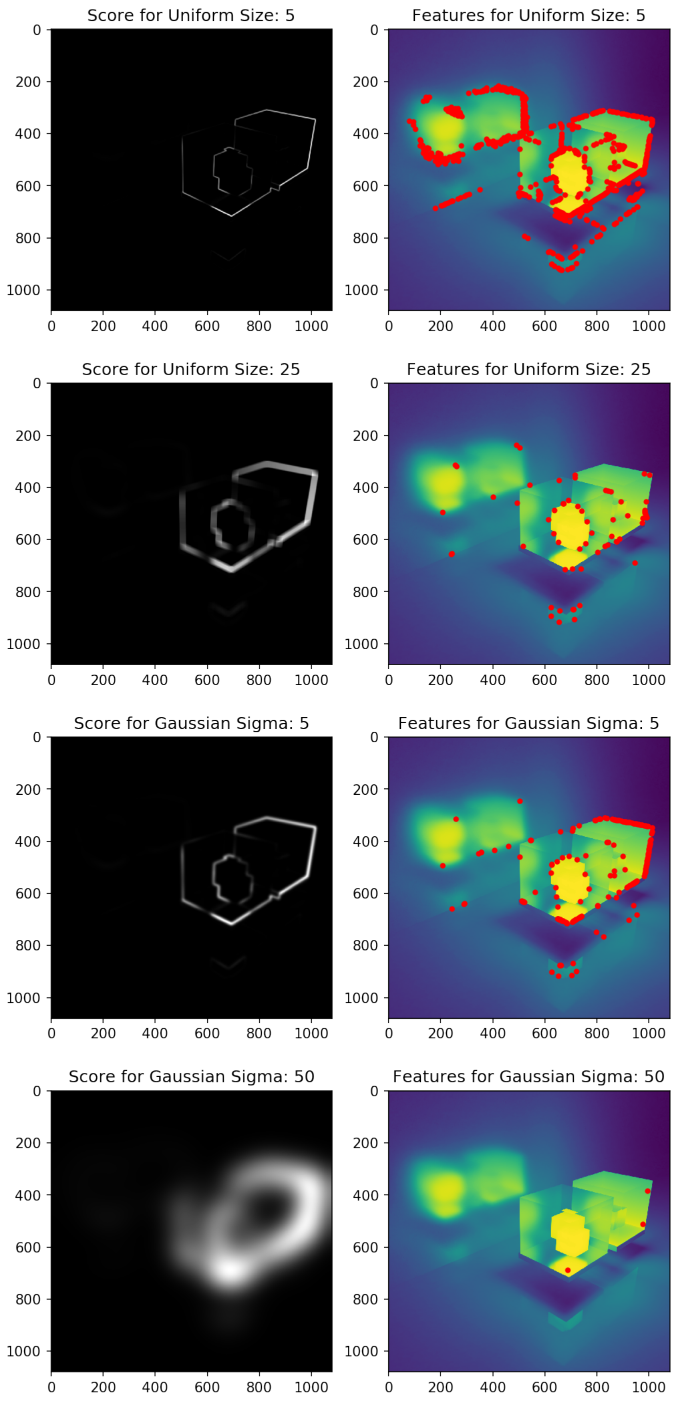

Fig. 3. Alternative Harris Scoring Function

2. **(GRAPH REQUIRED)** Plot "filter response" (the value of the image after the filter is applied) at the center of the circle versus σ. Confirm that the peak of the filter response at the center of the circle occurs at the σ we expect. (Recall that the peak σ value does not correspond to the radius of the circle).

QUESTION A: What is the relationship between the peak σ and the circle's radius?

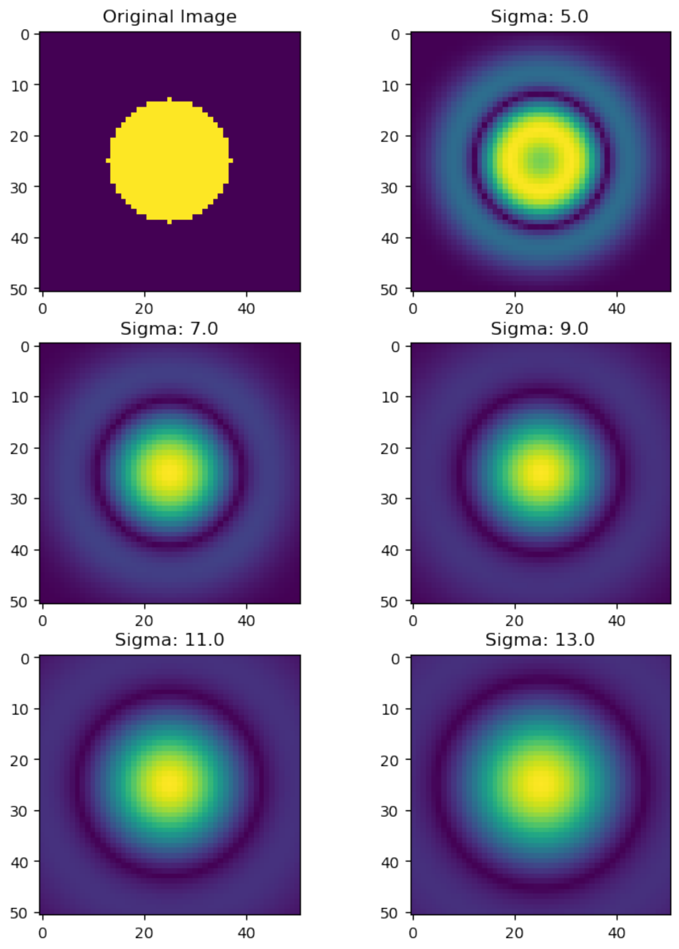

Fig. 4. Filter Response for Various LoG Sigmas

As you can see in Fig. 5. Max LoG Filter Response Curve, the maximum Response was at the sigma of 9. The relationship between the theoretical peak sigma and the circle's radius is as follows:

$$sigma = radius/\sqrt{2}$$

My circle's radius in this image is set at 12 so the expected maximum filter response would occur at

$$sigma = 12/\sqrt{2} = 8.485$$

which is quite close to our maximum tested value of 9.

B. P2.2.2 Annotating an Image with Multi-Scale Detections

Now, let's assume that we have an image with multiple features and we don't know either where they are or what their "radius" or "scale" is. Your goal here is to simultaneously detect features and estimate their characteristic scale.

For testing purposes, I have provided you with a simple image with two circles in it. Your task is to automatically identify where these "blobs" are and what their radius is. By the time you're done, you should be able to automatically detect the feature locations *and* their scale, producing images like the following:

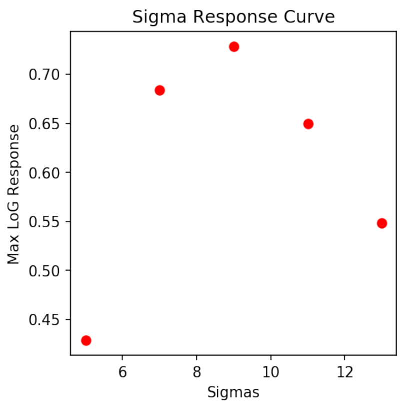

Fig. 5. Max LoG Filter Response Curve

Similar to the Harris Features from the last exercise, the features occur at extrema (both maxima and minima) in image-space. Before we compute multi-scale features, pick 3 or 4 values of σ and for each, plot the following:

1. **(GRAPH REQUIRED)** The filter response (applying the scaled LoG filter for a particular σ to the image function) and;
2. **(IMAGE REQUIRED)** The location of the extrema plotted on top of the original image (just like the Harris Corner exercise from the previous programming assignment).

Now, we can put everything together. The multi-scale features we care about exist at extrema in both image space *and* in scale space. This will require computing the "blob" feature response in both image-space and in scale space (by iterating through different sigma values). Features will exist at extrema of f in both image space and scale space.

Refer to Fig. 6. Multi Scale Filter Response for Various LoG Sigmas.

Once you have computed the features, plot them as circles of the appropriate radius on top of three images: (1) the two-circle "test" image I have provided, (2) either the "sunflower_field.jpg" image or the "light_cubes_base.png" image I have provided in this folder, and (3) an image of your choosing.

III. P2.3 Image Warping

In this question, we will pick up where we left off from the in-class breakout session. You are tasked with writing a `transform_image` function that takes as input an `image` and `transformation_matrix`. The function takes in an image and transforms it according to the `transformation_matrix`.

Using the `upsample_image` function I have provided below, write a new function `transform_image` that applies a transformation matrix to an image. You should feel free to use `scipy` for interpolation, **but you may not use `scipy` for anything other than the convolution, peak detection, and interpolation**.

Your goal, once this function is written, is to **implement the following transformation kernels, apply them to an**

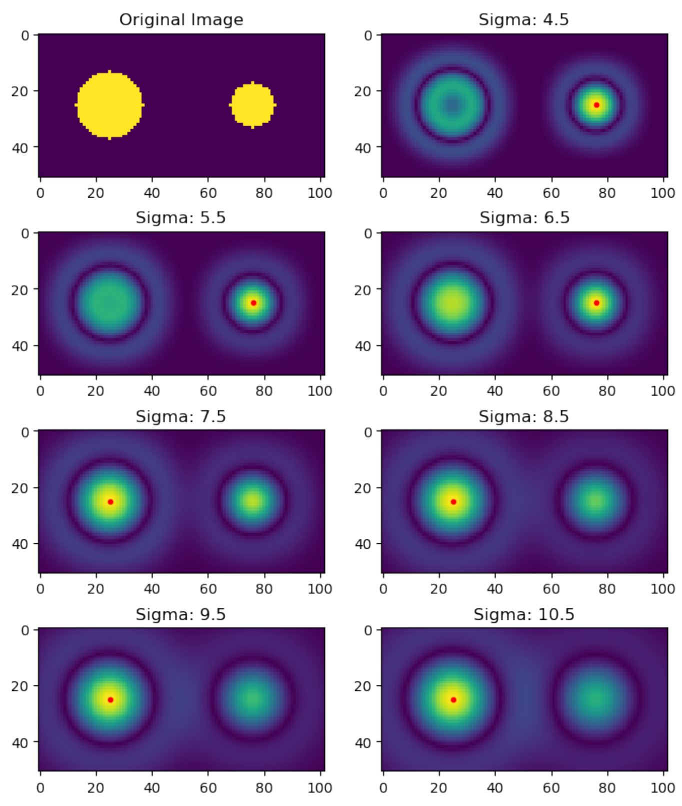

Fig. 6. Multi Scale Filter Response for Various LoG Sigmas

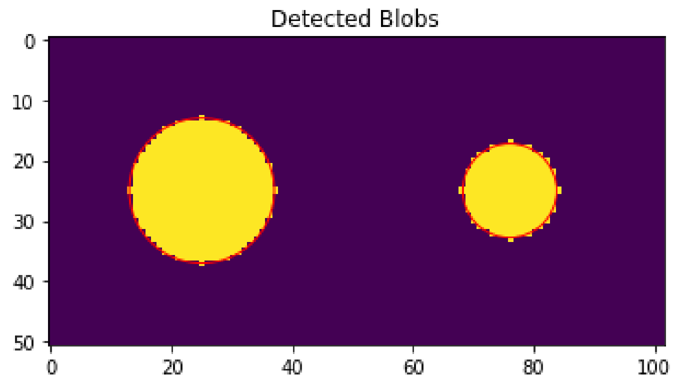

Fig. 7. Detected Blobs of radius 12 and 7.7 Respectively

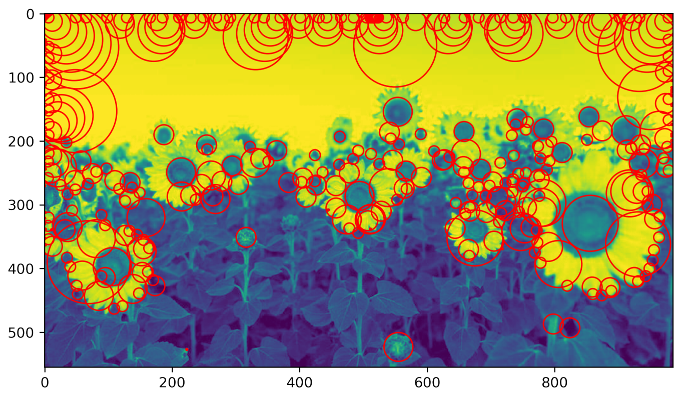

Fig. 8. Detected Blobs for a Sunflower Field

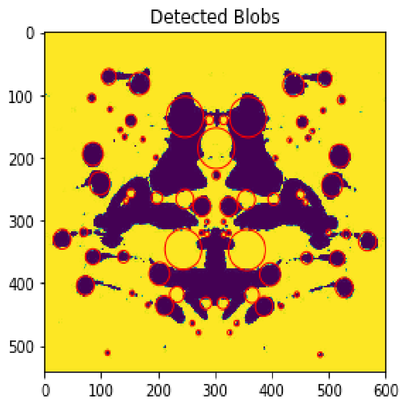

Fig. 9. Detected Blobs for a Rorscach Painting

image of your choosing, plot the image, and describe any surprising behavior of the kernels:

1. The identity:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. A rotation by 30 degrees

$$\begin{bmatrix} 0.8660254 & -0.5 & 0 \\ 0.5 & 0.8660254 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3. A rotation by 30 degrees and translated to the center

$$\begin{bmatrix} 0.8660254 & -0.5 & 80.83176102 \\ 0.5 & 0.8660254 & -46.66823898 \\ 0 & 0 & 1 \end{bmatrix}$$

After centering, the image is larger than the original because the corners of the original image have to be included in the image as well so each dimension grows to account for it.

4. Scale by a factor of 2 along the x-axis

$$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5. This kernel (which you should describe):

$$\begin{bmatrix} 1 & -1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This kernel shears the right side of the image down and bottom of the image left the same distance as the size of the image. This is similar to rotating by 45 degrees and and doubling the size of the image but by expanding in one direction along each axis rather than expanding from the center of the image.

QUESTION A: How do the changes in these vectors correspond to changes in the warped image?

The warped image such as in scale x by 2, is shown to be stretched along the x axis and widened.

Finally, experiment with homography transforms, by modifying the bottom row of the transformation matrix. Define two different homography transforms, write out their matrices and display the results.

1) Homography 1

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.006 & 0 & 1 \end{bmatrix}$$

2) Homography 2

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.006 & 1 \end{bmatrix}$$

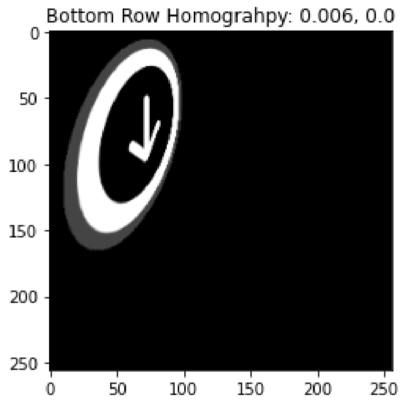

Fig. 10. Applied Homography 1

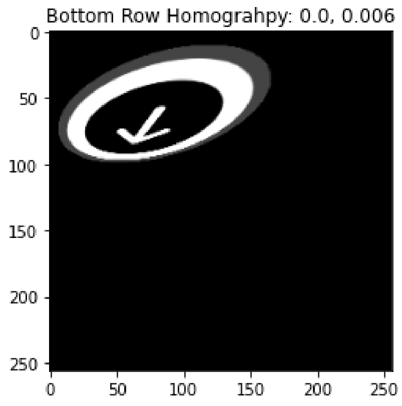

Fig. 11. Applied Homography 2

QUESTION B: How do the two "bottom row" parameters control how the image is warped?

The bottom two parameters control how the image is projected from another plane. The higher each number gets, the more rotated the image gets (to face left/down for the bottom left or up/right for the bottom middle instead of directly at you). This can be shown in Figure 10 and 11. In Figure 10, the image has been rotated in space in the x direction to have the right side of the image appear to be farther away. Similarly, in Figure 11, the second homography rotated the image in the y direction with the top of the image appearing closer and the bottom rotated away from the viewer.

IV. P2.4 Implementing Some Simple Feature Descriptors

Compute some of the simple image descriptors we discussed in class and see how effective they are at matching features across different image transformations. To locate the features, use your Harris Corner detector from question P2.1. I have provided code that computes the feature descriptors for small image patches surrounding each of your features using three of these strategies. I have also included code under `Descriptor Matching Plotting Code` that plots the matches for a set of test images.

TASK Implement the `get_corners` function (in the code block named `Computing and Matching Descriptors` using your Harris Corner detection solution from P2.1. If that function is implemented properly, you should be able to generate plots showing match quality.

FIGURES Once you have implemented `get_corners` the code below should generate plots for the three different descriptors. Include all three in your writeup.

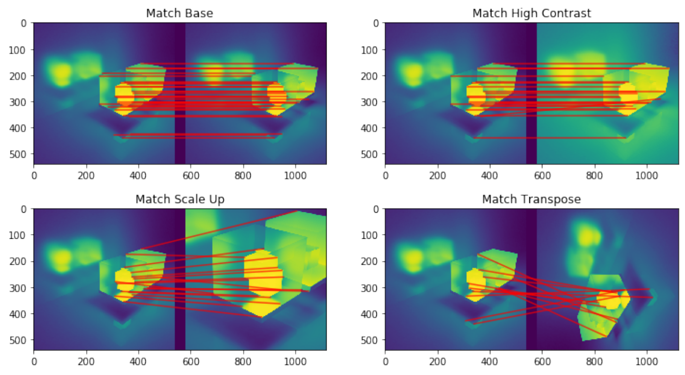

Fig. 12. Match Feature Matching

QUESTION A Which feature descriptor performs poorly on the image `img_contrast`? Explain why this descriptor performs worse than the others.

Histogram performs much worse on the contrast image because it relies on a histogram of colors. This is resistant to spatial transformations but if the colors change via a transform such as increasing contrast, the histograms will no longer match at all for the same feature.

QUESTION B Which feature descriptor performs best on the image `img_transpose`? Explain why this descriptor performs better than the others.

Histogram performs the best on the transposed image because the histogram preserves the local relationships between

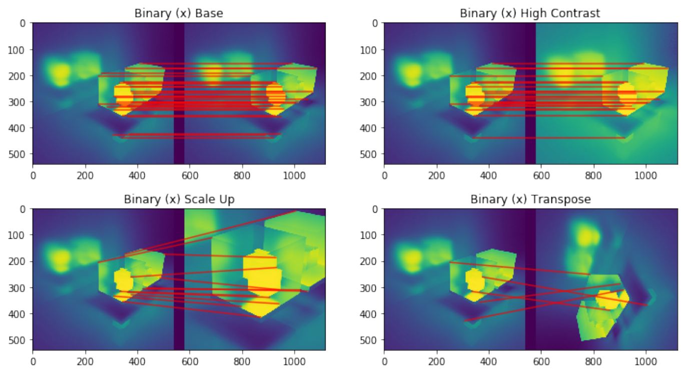

Fig. 13. Binary Feature Matching

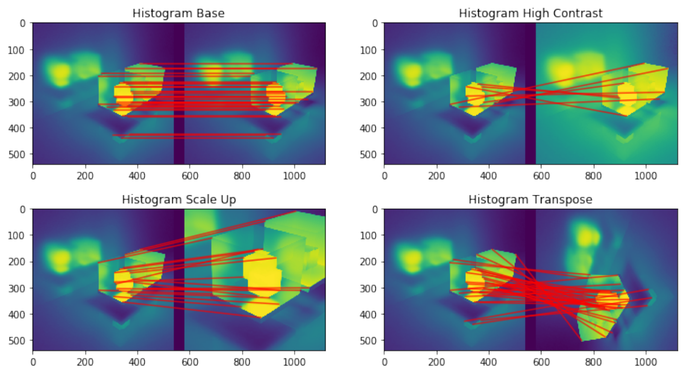

Fig. 14. Histogram Feature Matching

neighboring pixel colors. This is partially resistant to orientation changes so transposing the image does not impact the performance. Match finds many matches in the transposed image but if you look closely, they are not correct matches. It finds similarity between various corners in the two images because a transpose of a corner still is a corner but with an orientation change. This means that different corners will match when one is transposed.