EECS 490

Assignment 2

Warping and Registration

Edward Venator
Due: 9/20/2011
Handed In: 9/21/2011

This report examines the results of registering images to templates using an affine transform with three control points with MatLab's image processing toolbox.

Technical Discussion

It is a common task in image processing to compare one image to another. However, the images must be aligned for this to be possible. Images may be misaligned due to camera movement, movement of the image subject, or two image subjects being slightly different.

The process of aligning to misaligned images is called image registration. The process of image registration consists of defining a series of control points in the template image and the corresponding control points in the image to be registered. From this point-to-point mapping, it is possible to create a transformation matrix. The transformation matrix can then be used to remap the input image's pixels, aligning it to the template image.

The actual transform can be accomplished with a variety of methods. In this assignment, I used an affine transform, which uses a combination of shear, scaling, and rotation. It is ineffective for images that have local distortion, but is suitable for skewed, scaled, or rotated two-dimensional objects, such as the patterns in these images. The affine transformation matrix has six terms, so at least three control points are needed in each image to solve for the transformation matrix. MatLab uses pseudoinverse techniques that allow the matrix to be defined using more than three control points, but I specified only three per image. The transform itself can be accomplished using forward mapping from the input image to the output image space, but this causes some input pixels to be mapped to the same output pixel and some output pixels to be left empty. MatLab uses reverse mapping from the output image space to the source image. This requires some kind of interpolation to determine the best value for output pixels, since most will map to spaces between input pixels. I chose to use bilinear interpolation, but MatLab allows any number of interpolation methods.

Results

The following is an image-by-image discussion of my results. For each image, I show the chosen registration points, the resulting image, and a subtraction of the output image from the template image (both converted to grayscale).

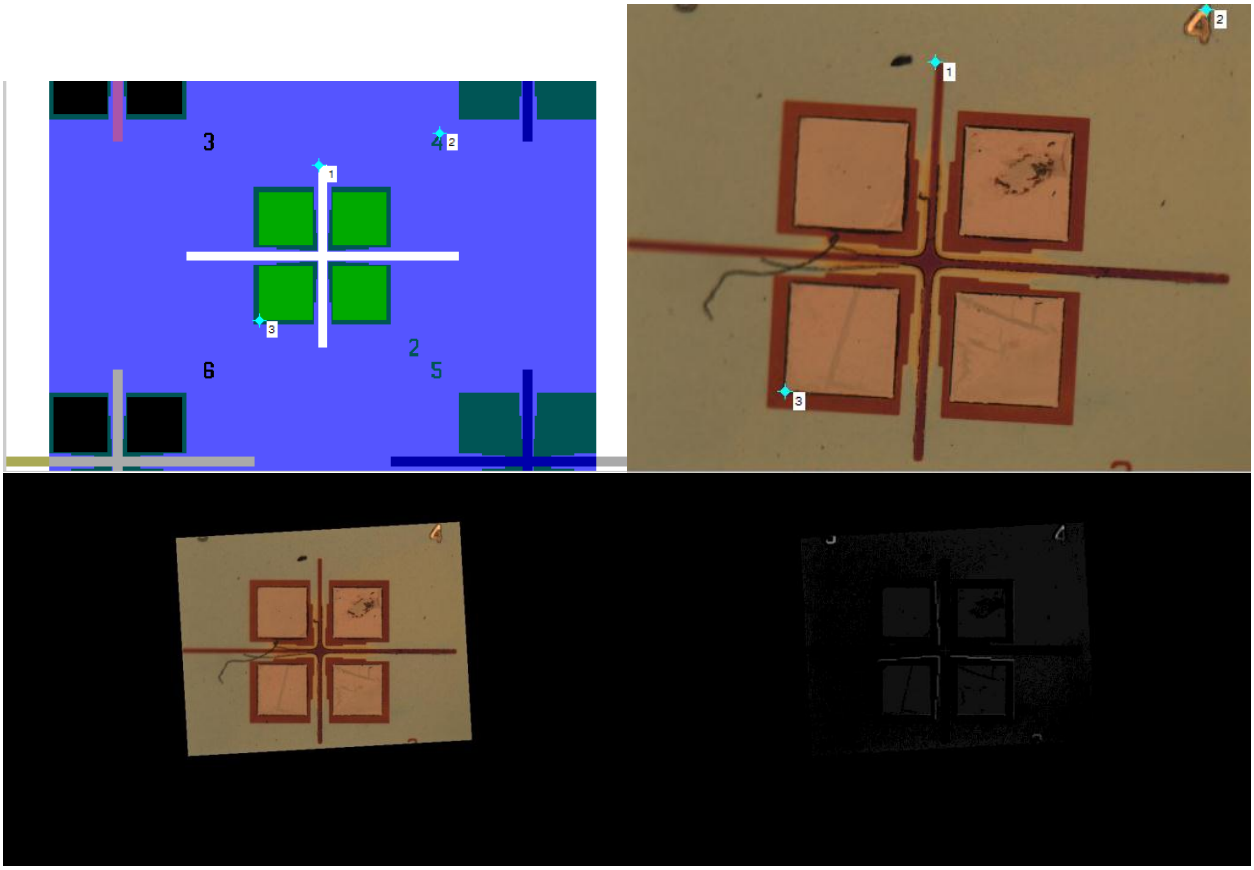

Figure 2: cantilever_layout1.bmp and cantilever1_2.jpg

In this image, I chose registration points that were easy to pick out but as close to the edges as possible. As you can see in the subtracted image, the template did not line up perfectly with the original image. However, most of the differences in the image are from actual differences between the template and the fabricated device, such as dust, scratches, and the limits of the manufacturing tolerances.

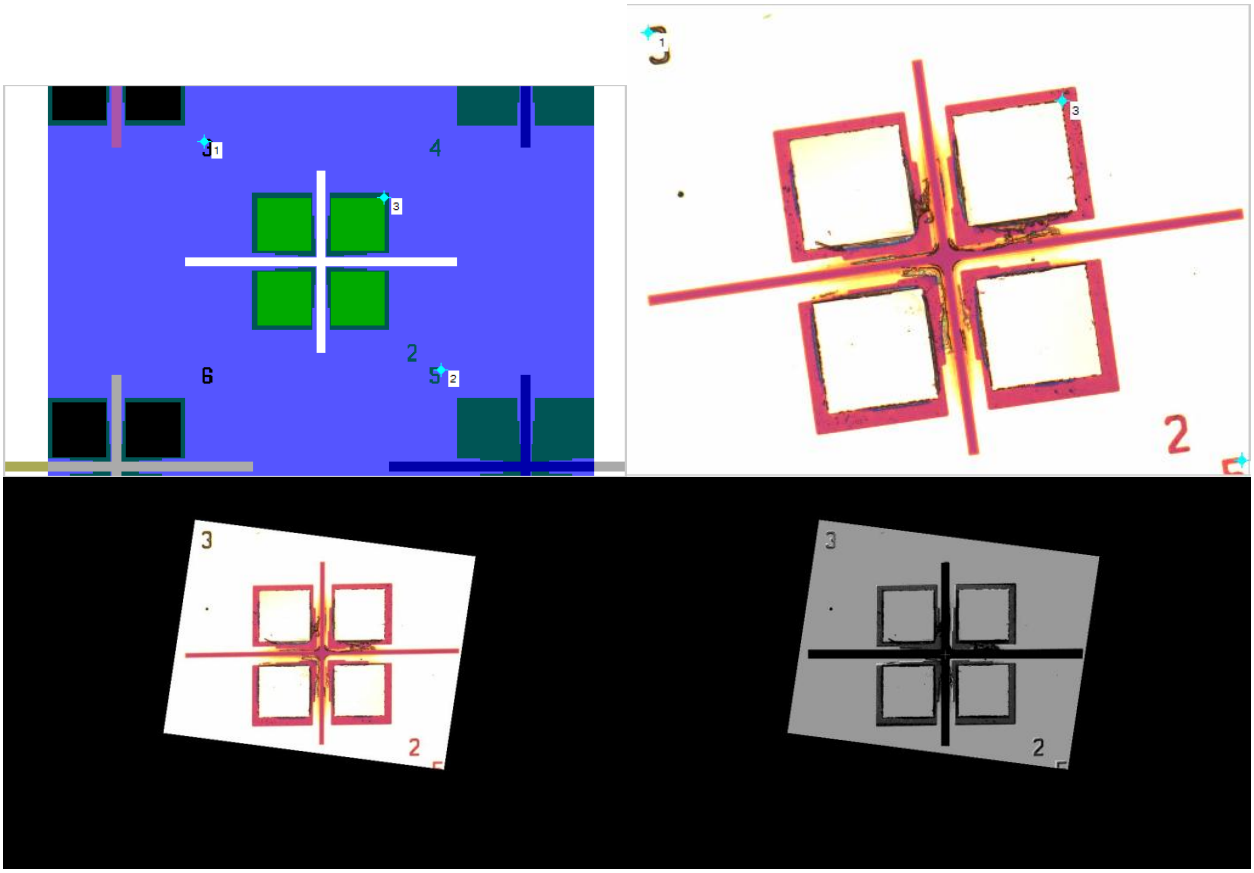

Figure 1: cantilever_layout1.bmp and cantilever1_1.jpg

This image registered very well. I used registration points very near to the edges of the images. In the comparison subtraction, you can clearly see the corrosion or burn marks in the image, as well as the specs of dust and manufacturing defects. This image pair also illustrates the effects of having different colors in the image and template. Even though the template and image line up, the output image shows the pattern in shades of grey and black instead of all black for a perfect match. This is because the colors in the image do not match the colors in the template.;

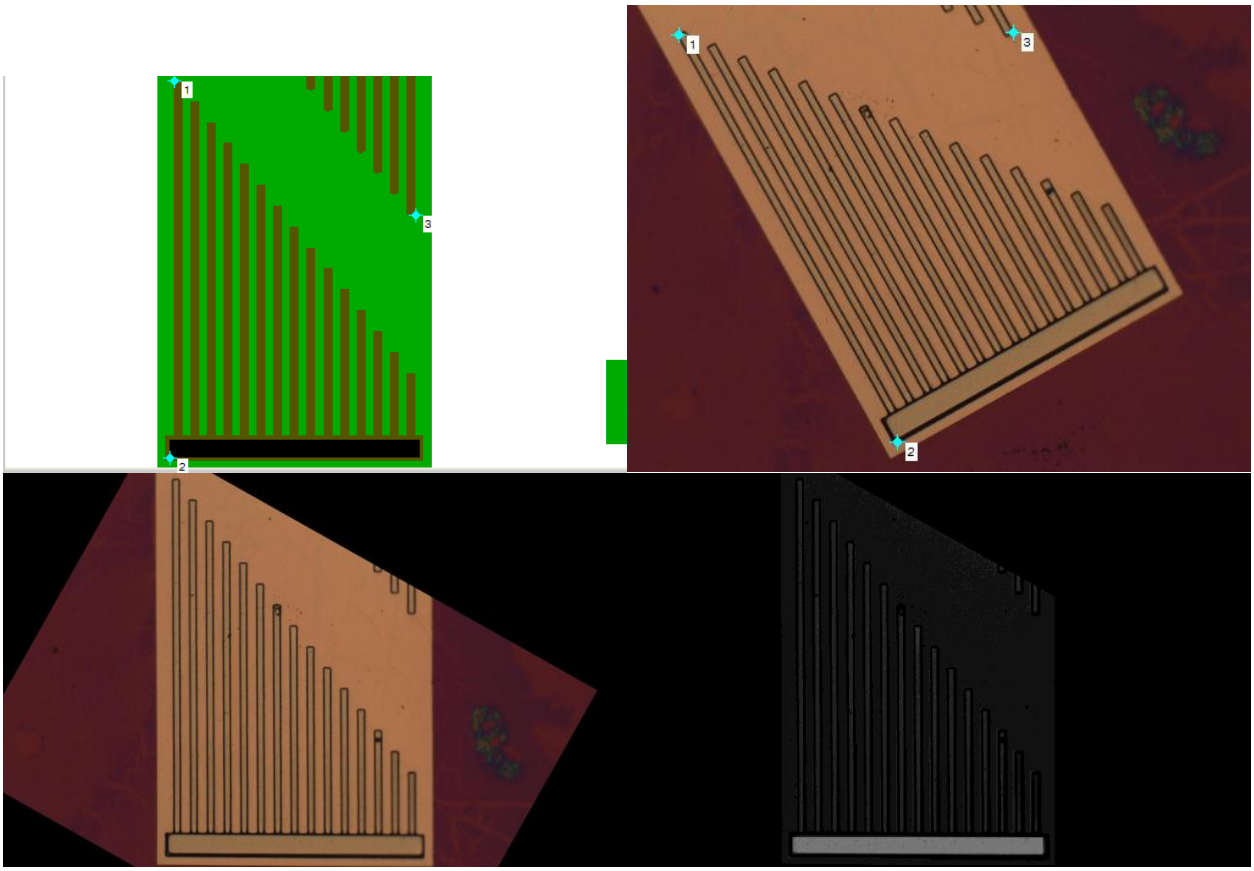

Figure 3: cantilever_layout2.bmp and cantilever2_1.jpg

This image was very easy to align because it has many geometric features from which to select control points. The output comparison shows how the manufactured part is slightly embossed, resulting in a “shadow” outline of the features, whereas the original has no lines along the edges of the features.

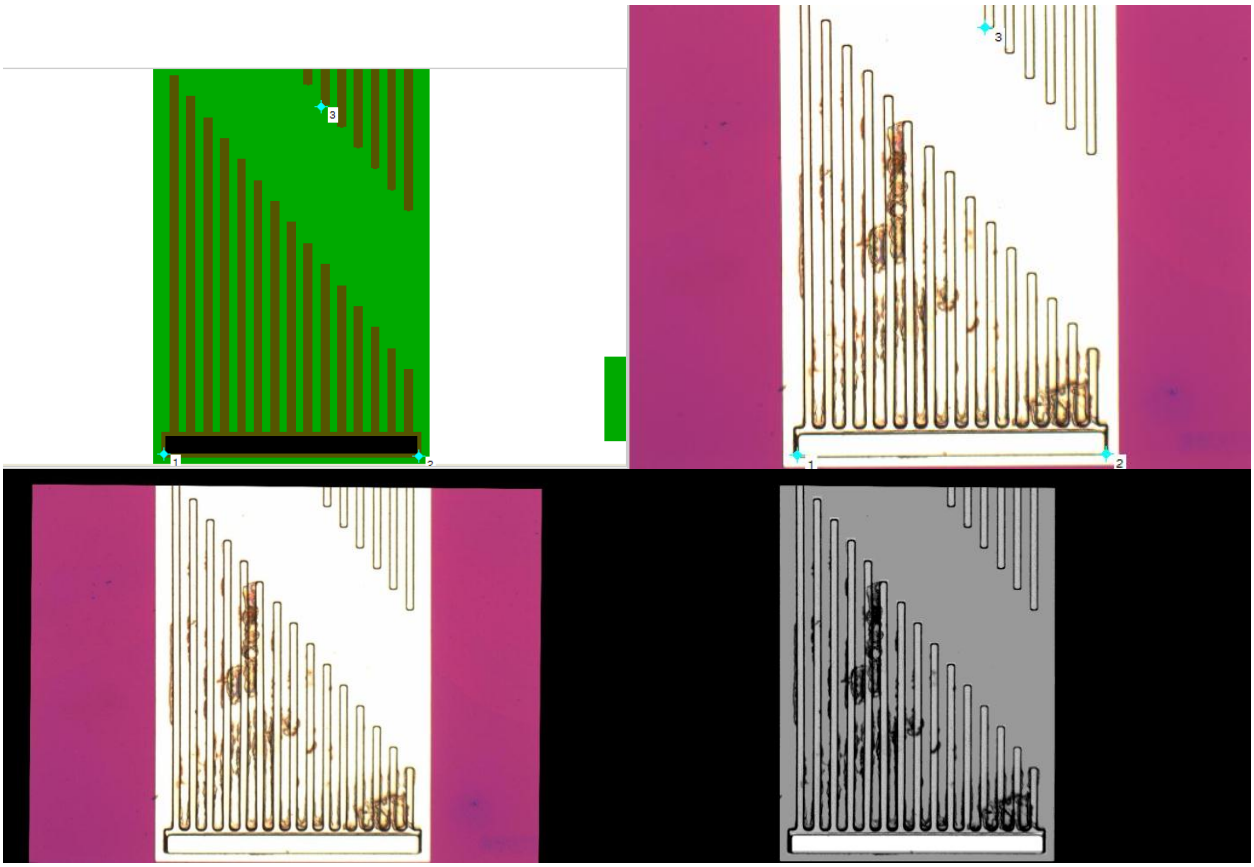

Figure 4: cantilever_layout2.bmp and cantilever2_2.jpg

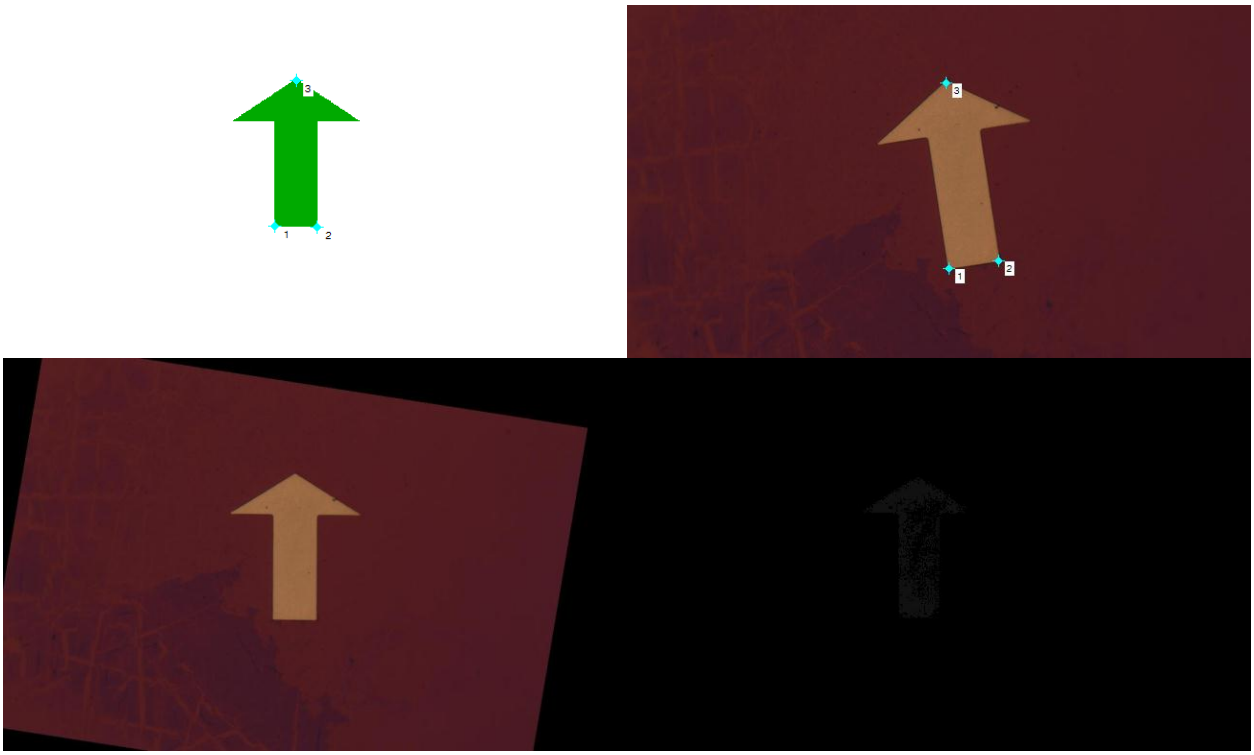

Figure 5: cantilever_layout3.bmp and cantilever3_1.jpg

This image has very simple geometry. Although I chose the control points near the center of the image, the only important feature is the arrow, which registered very well. The output image shows that there is almost no difference between the template and image. There are no telltale outlines to indicate poor alignment, only some dust.

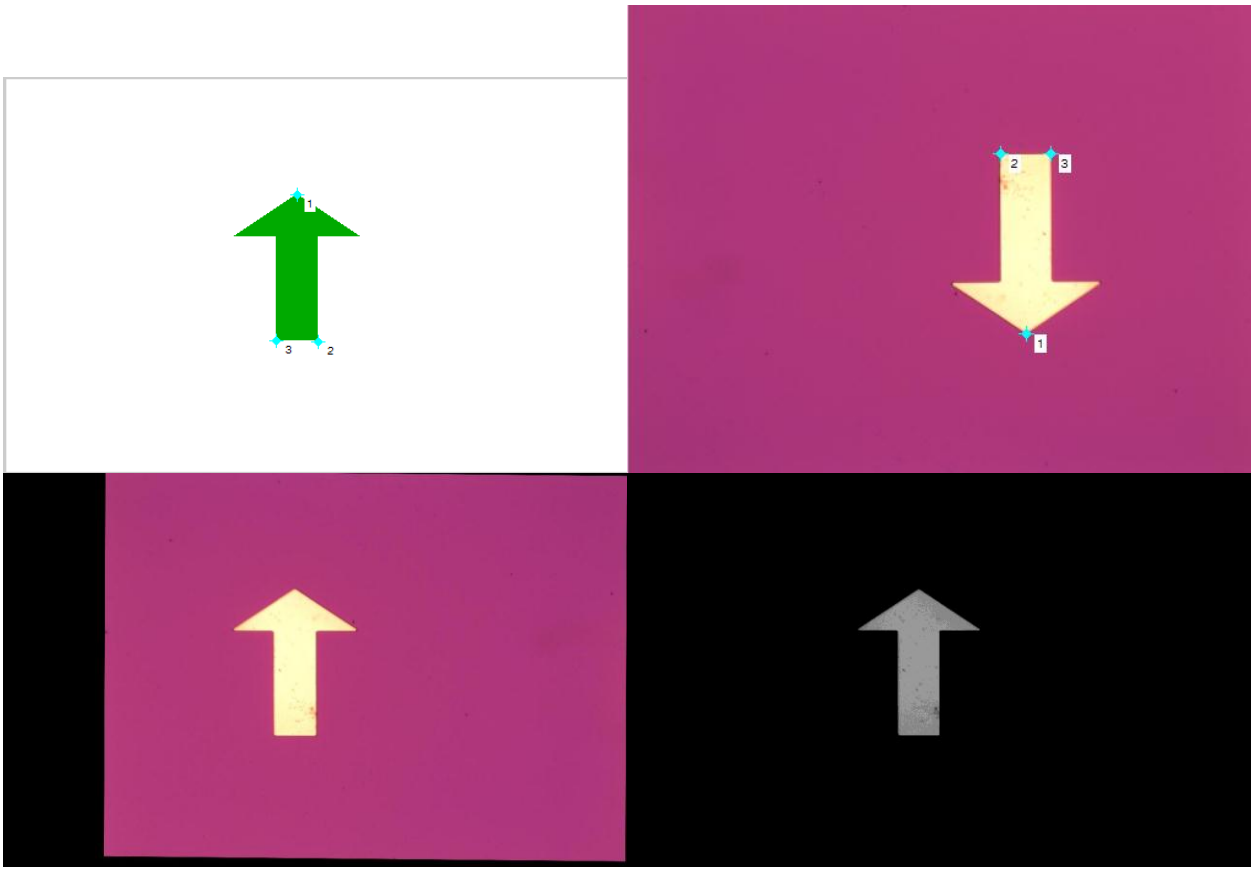

Figure 6: cantilever_layout3.bmp and cantilever3_2.jpg

Once again, this image registered very well, in spite of having to perform a 180° rotation. The output image clearly shows the dust on the input image.

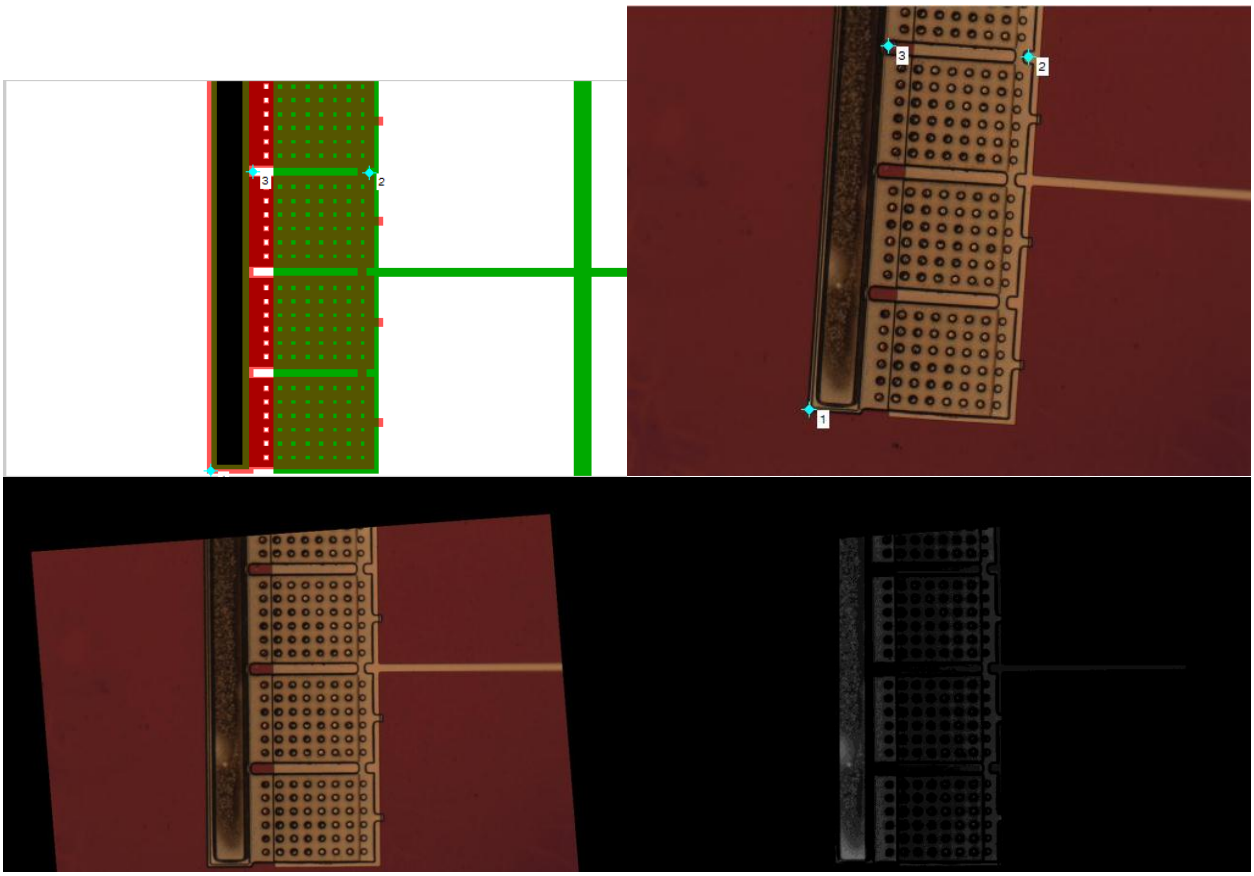

Figure 7: cantilever_layout4.bmp and cantilever4_1.jpg

Although this image registered very well, the output shows several artifacts that are the result of the template geometry not matching the image geometry. For example, the template has square holes, but the image shows round holes, probably because of manufacturing limitations. The output does very clearly show the corrosion (or whatever it is) on the left side of the part.

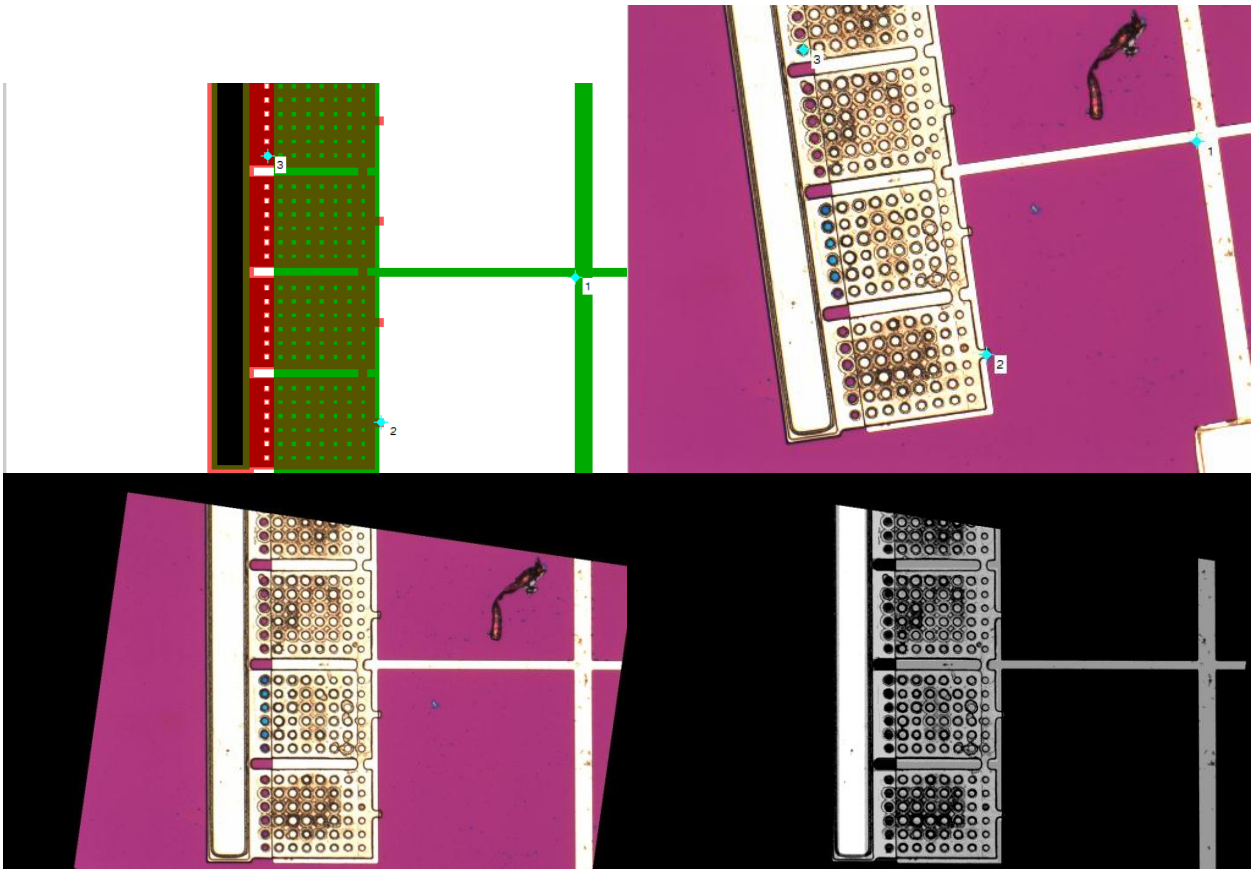

Figure 8: cantilever_layout4.bmp and cantilever4_2.jpg

This image appears to have registered very well, probably because the three points were so far apart. The output clearly shows imperfections in the manufactured part. However, because I did not cast the images to signed integers before subtracting, some imperfections are not visible in the output (because the difference in the images was negative and thus zero).

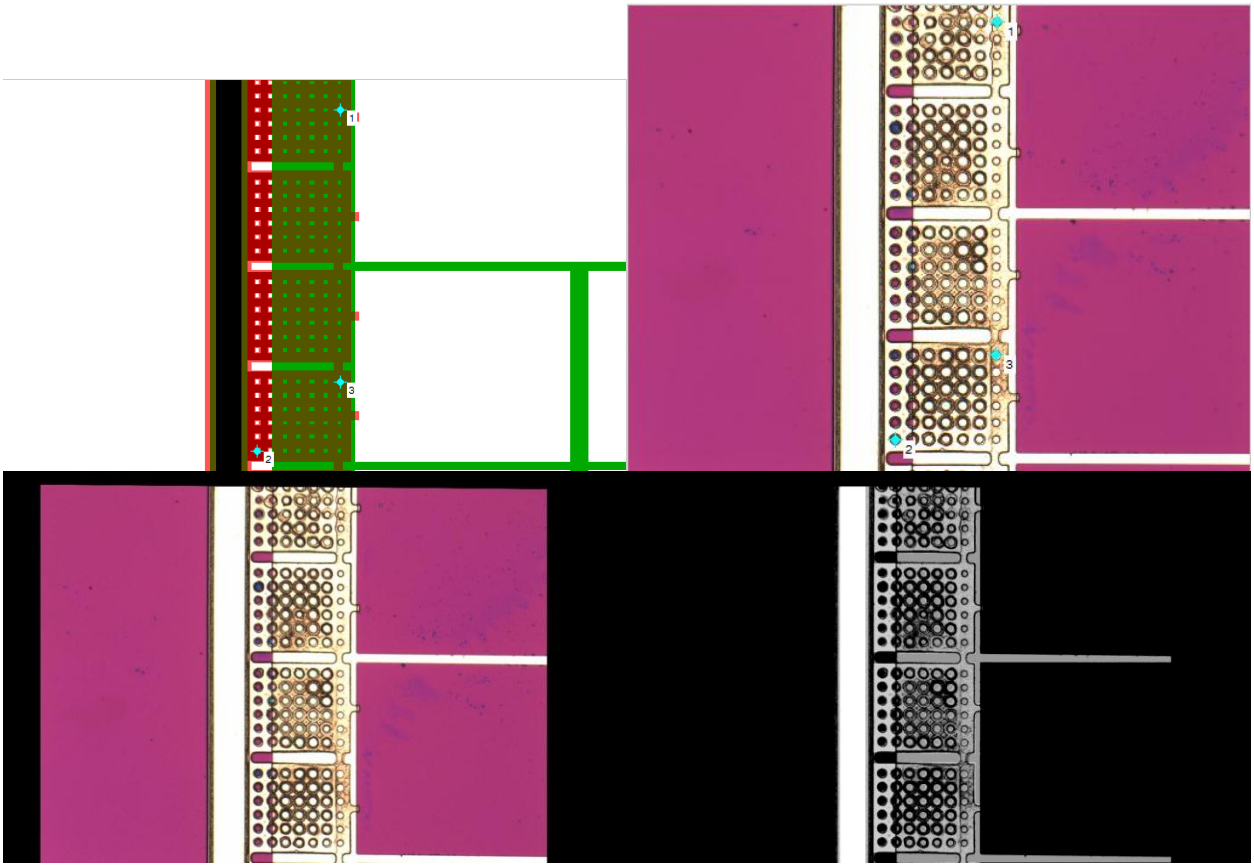

Figure 9: cantilever_layout5.bmp and cantilever5.jpg

In this image, I started using a new method of selecting registration points. I noticed that the small markings for the holes were a very precise way to align my points. As you can see in the output, this is an effective method of selecting points and ensuring that they match up properly.

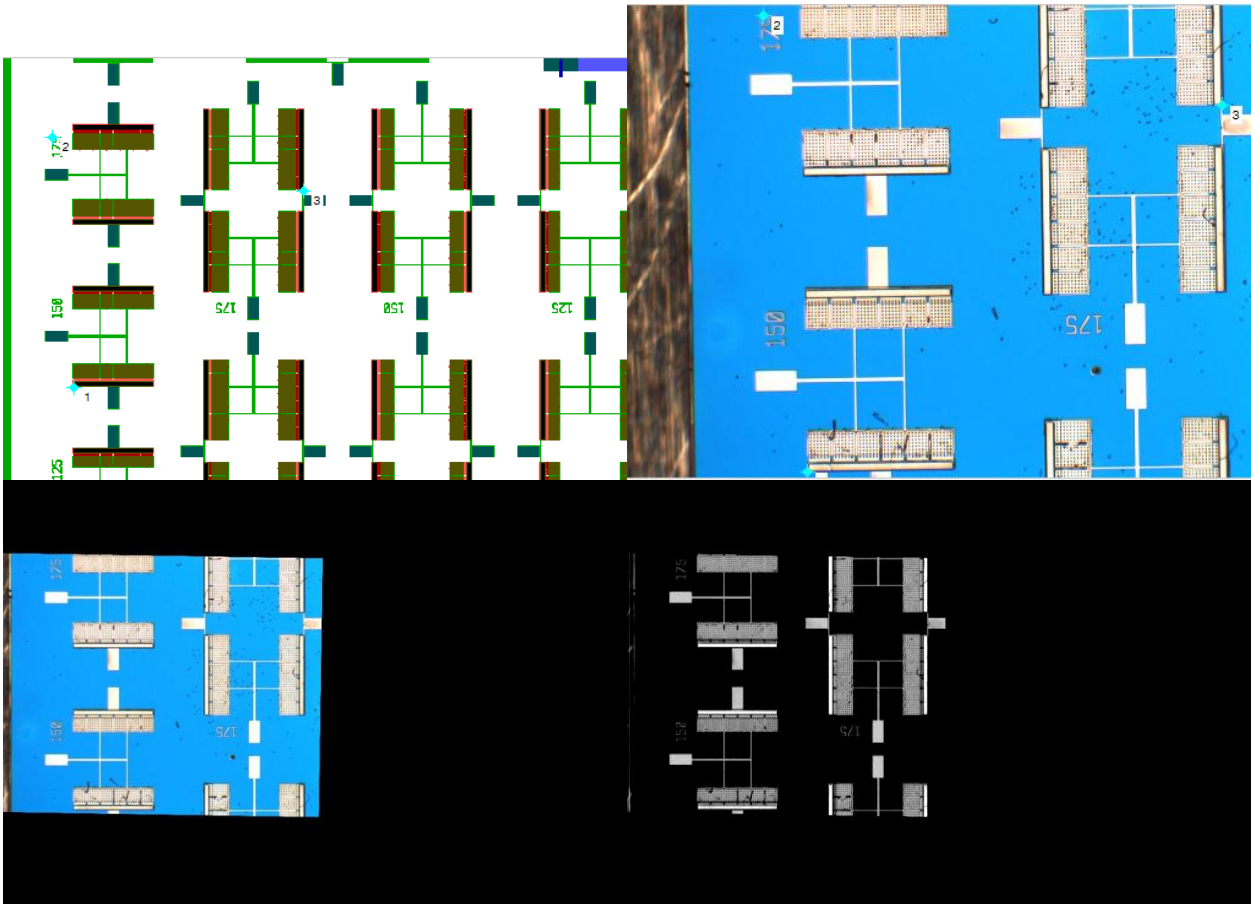

Figure 10: cantilever_layout6_1.bmp and cantilever6.jpg

This image was only a fraction of the size of the template, but it was still able to register well. The ouput image shows dust and imperfections on the manufactured part.

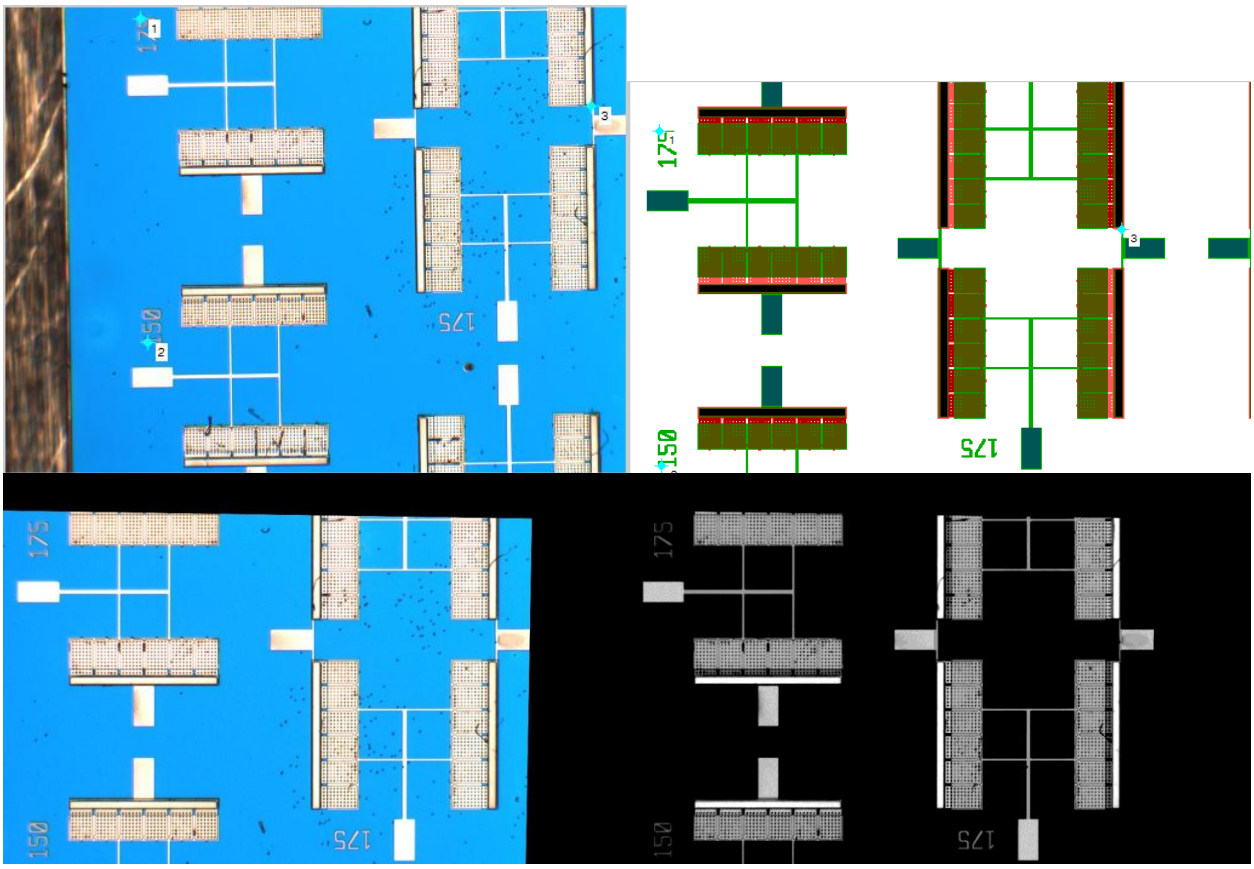

Figure 11: cantilever_layout6_2.bmp and cantilever6.jpg

This image, despite registering well, again demonstrates the problem with my subtraction method: because I was subtracting the template from the transformed image, the white of the template effectively “masks” any imperfections in the image.

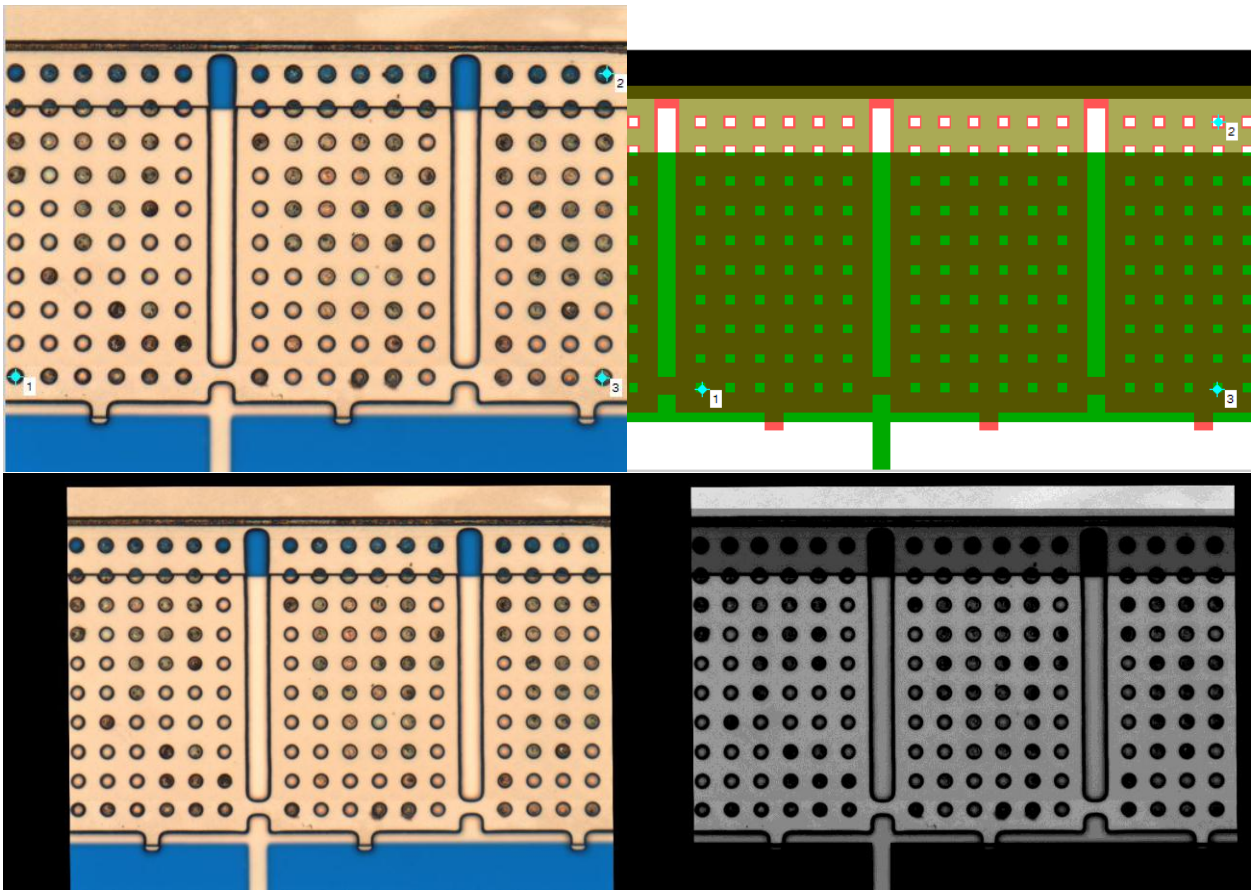

Figure 12: cantilever_layout7.bmp and cantilever7.jpg

This image does not appear to have registered very well. The vertical lines in the template remain slightly skewed in the output. I am not sure what causes this, but it may be that my points do not match up perfectly.

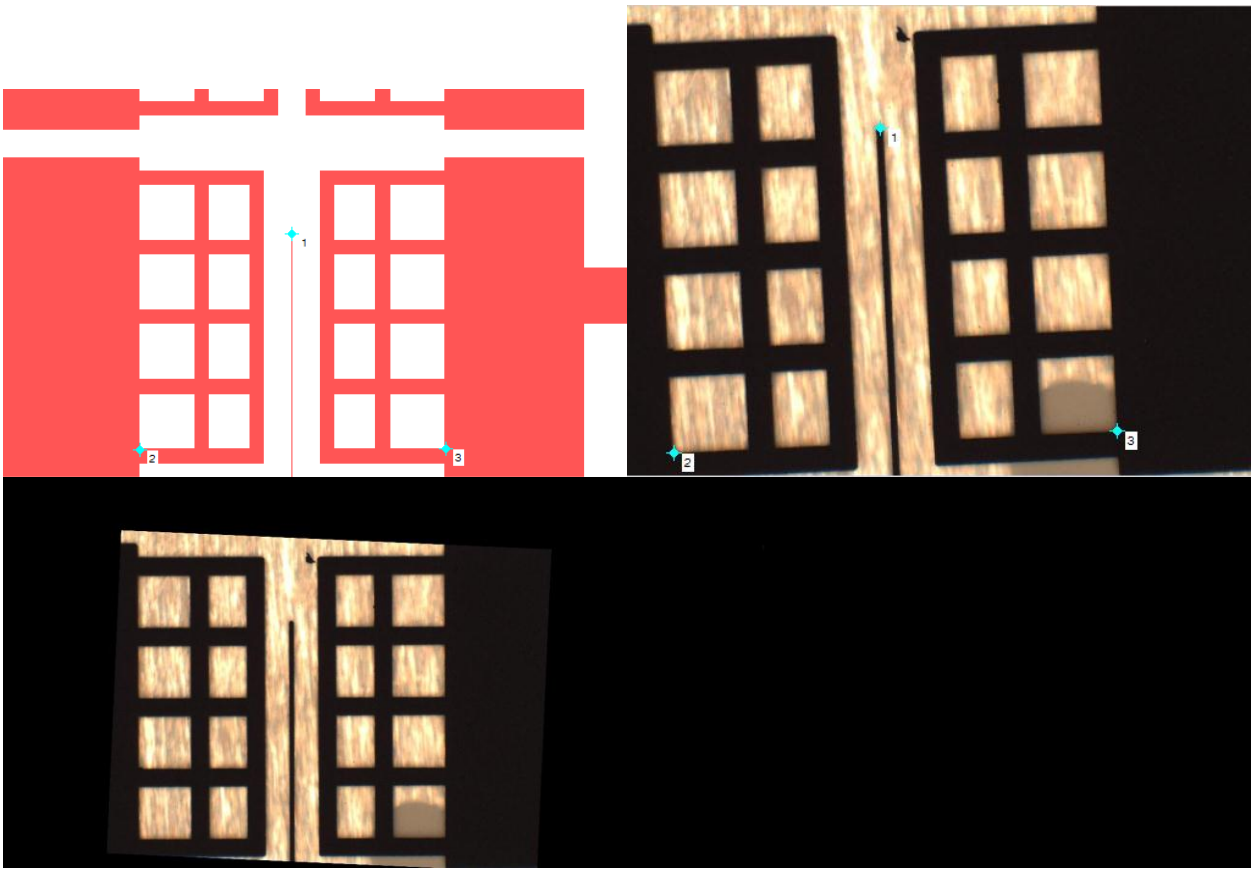

Figure 13: corona_layout1.bmp and corona1.jpg

This image looks like it registered very well, but my subtraction comparison method does not give any information on the imperfections in the image because the white template masks the pattern of the metal.

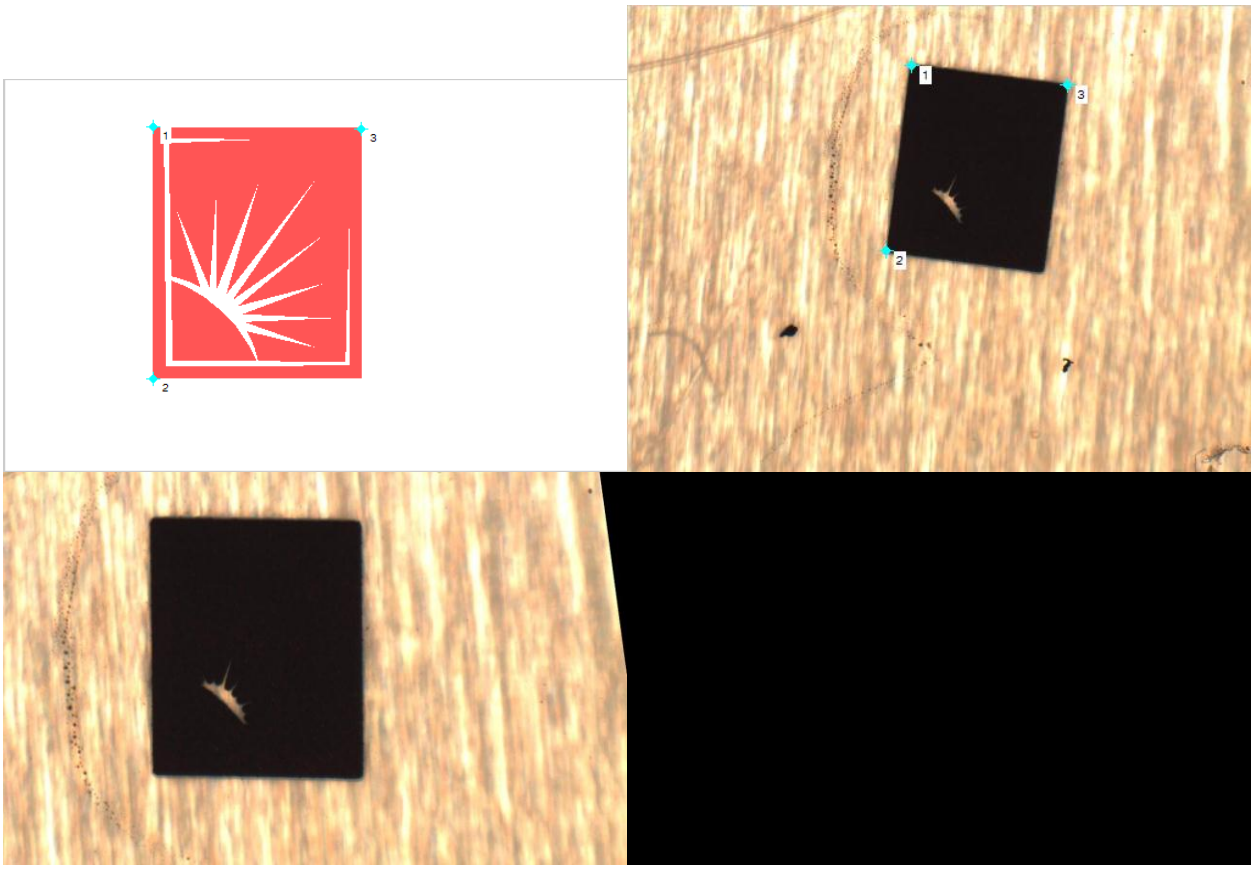

Figure 14: corona_layout2.bmp and corona2.jpg

Again, this image looks like it registered very well, but my subtraction comparison method masks the pattern of the metal and the markings on the metal.

Appendix

Scripts

process_image.m – Defines a function that takes in the template file name and comparison image file name (without file extensions), prompts the user for control points, and generates the output. The transformed image and comparison image are saved in /output.

main.m – Processes all of the images in /pics

Images

Pics directory – All of the input images provided with the assignment

Output directory – All of the transformed images and subtracted comparison images shown in this report.

Data Files

Output directory – All of the configuration points used in this report. These are saved over when the user selects new points in the point picker window.