Project # 2: Local Feature Extraction, Detection, and Matching

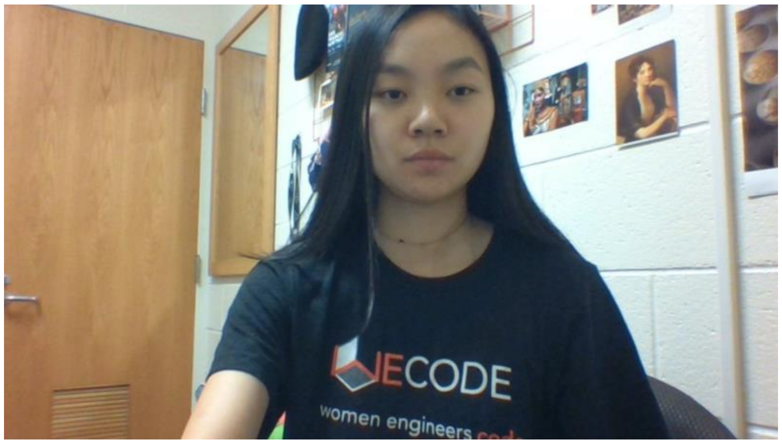
CSC 391: Computer Vision

Caroline Li

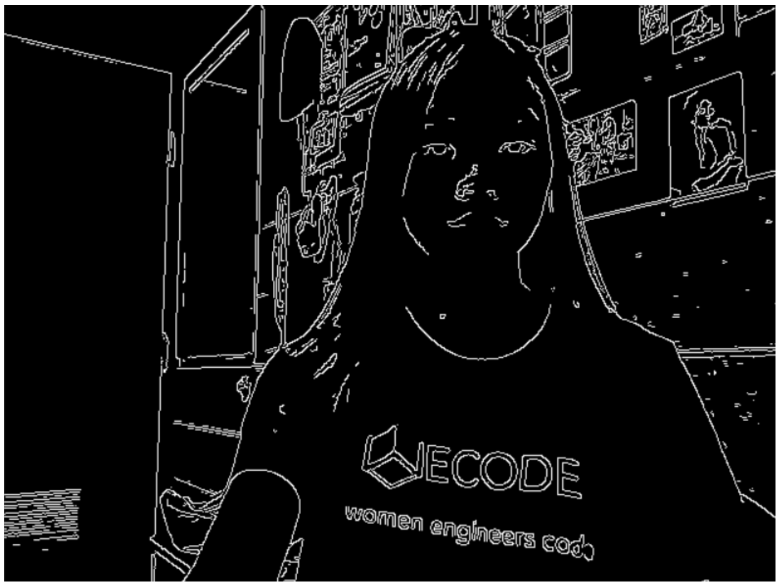
February 26, 2019

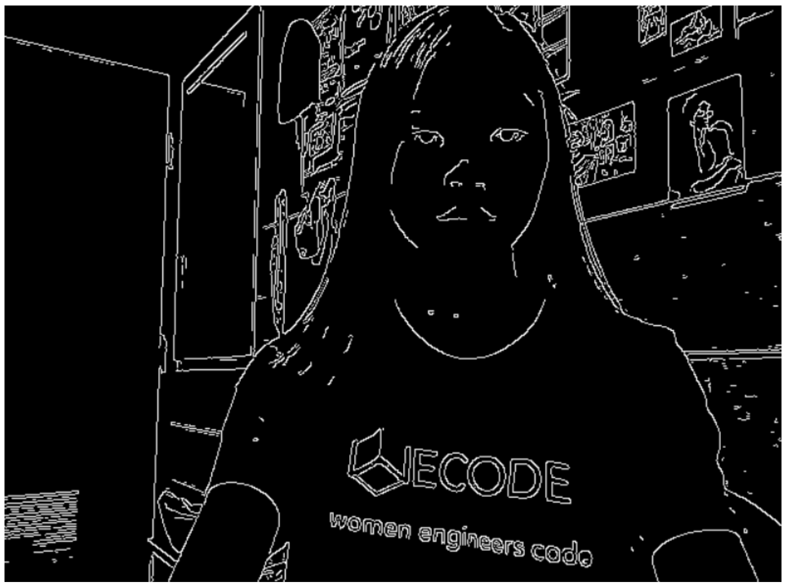
1 Experimenting with edge detection

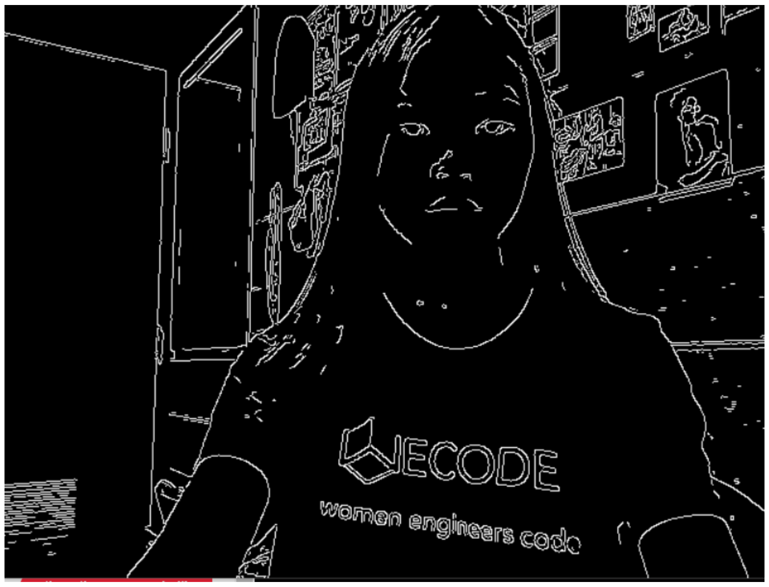
Using OpenCV's Canny edge detection function, I played around with minimum and maximum values to threshold which edges would be kept based on their intensity gradient in comparison to these set values.



Normal Lighting

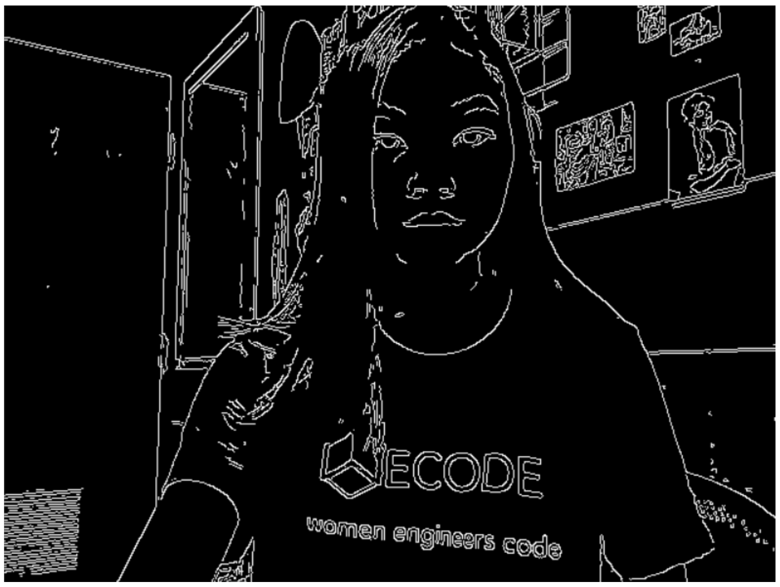


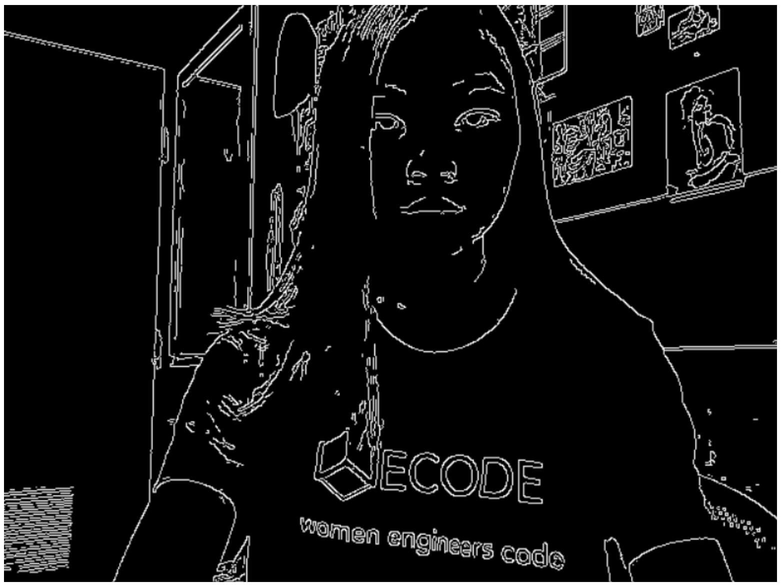


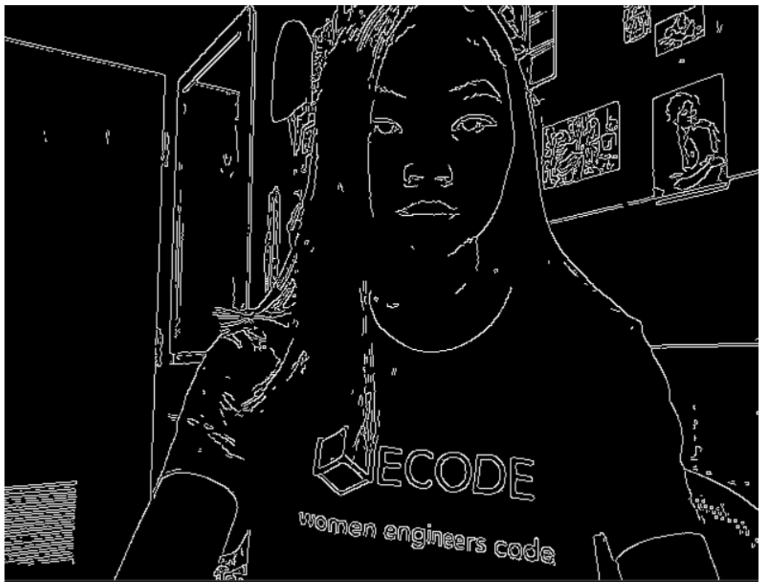


Min 100, Max 150

With the minimum value set at 100 and maximum value set at 150, edge detection is able to detect edges both in the foreground in the background very well. It's clear where the door, mirror, and postcards on the wall are, as well as myself, my facial features, and the design on my shirt.

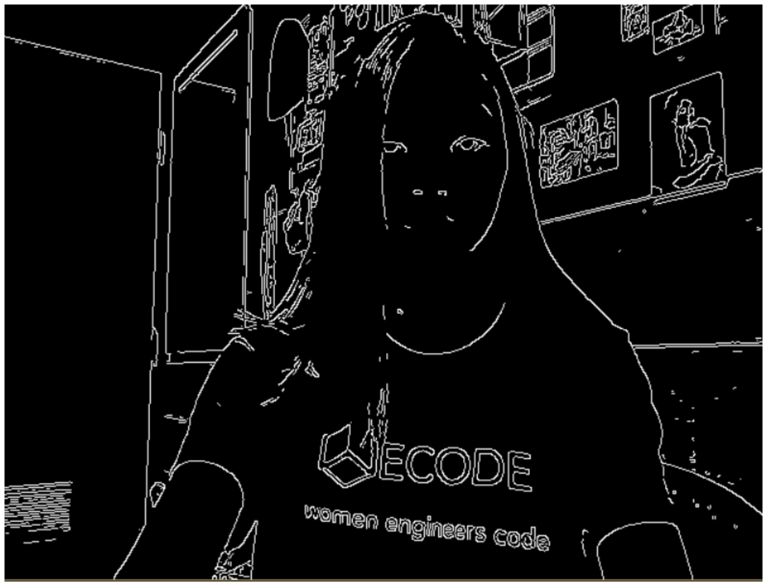

ECODE
women engineers code

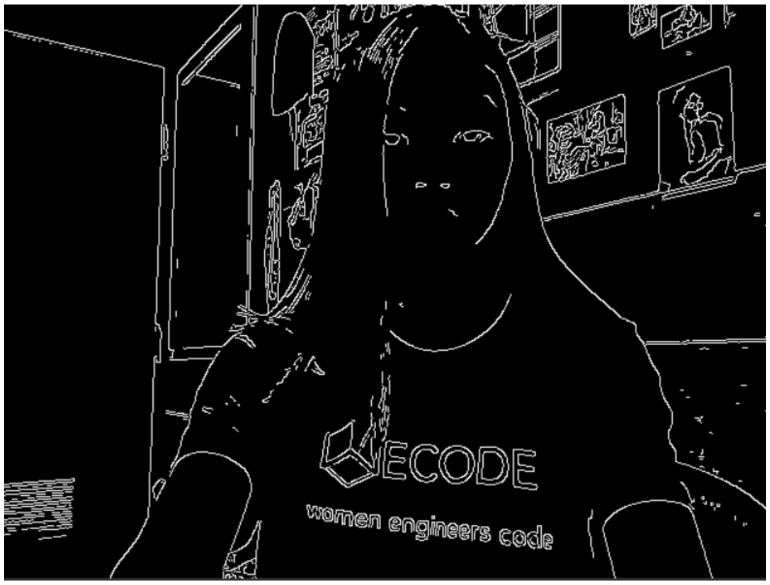

ECODE
women engineers code

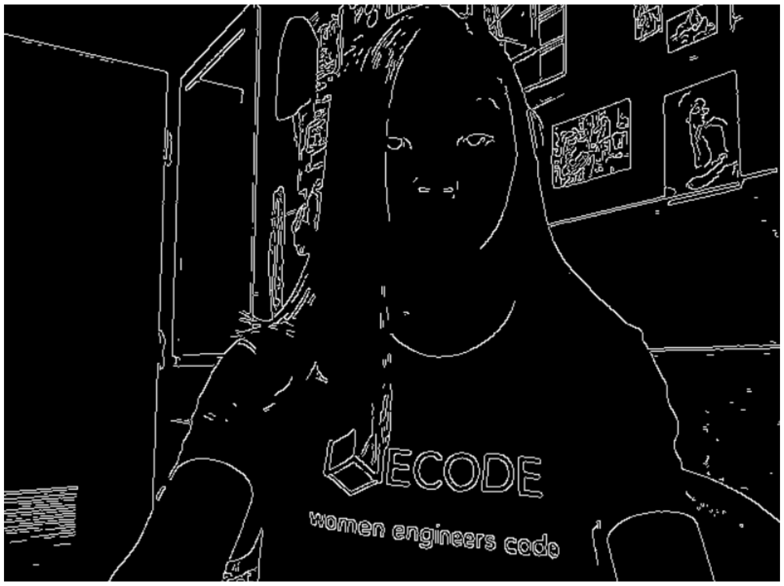


Min 75, Max 125

Shifting the minimum and maximum values down by 25, more features in my face show up, however you can see some discontinuities in image 1 and 3 on the door, for example.





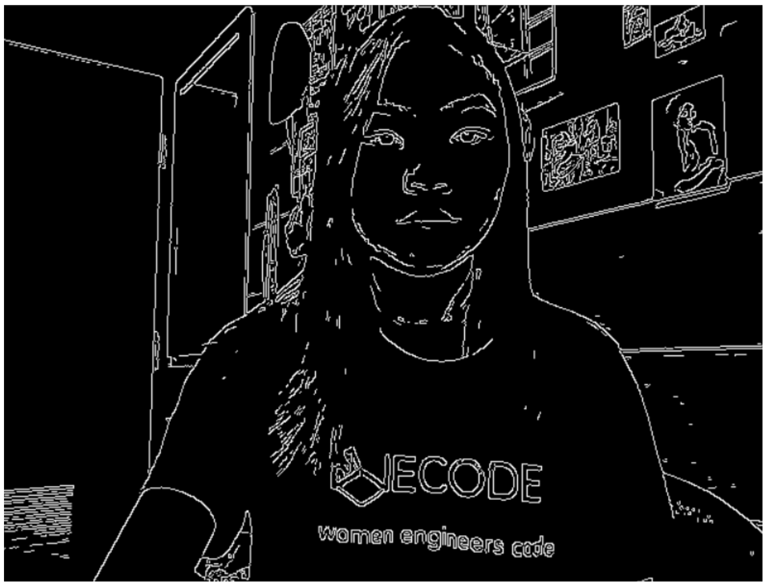


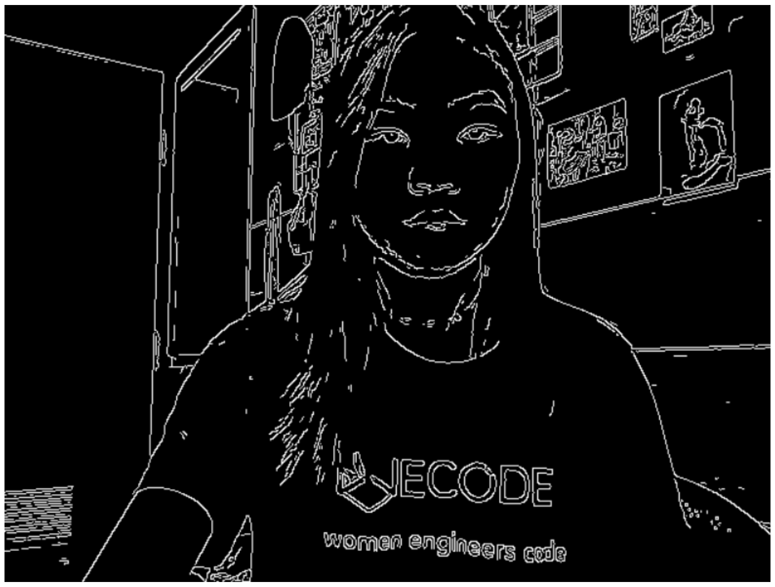
Min 125, Max 175

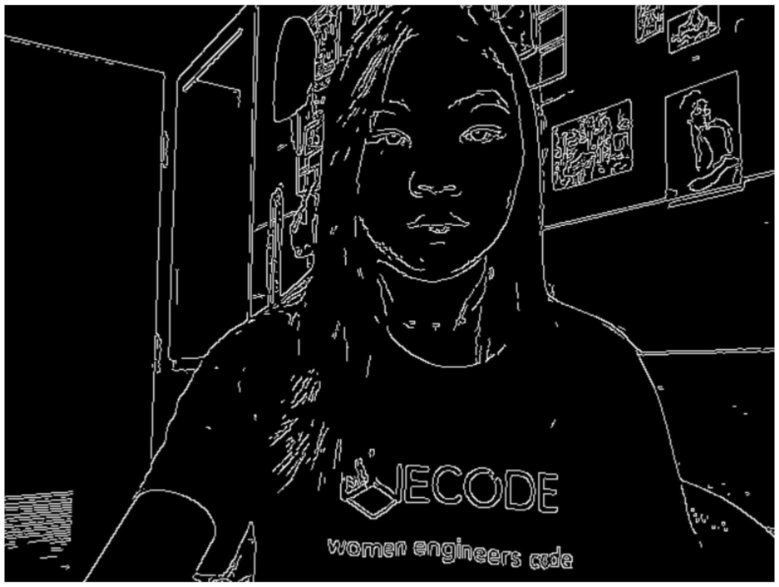
With values shifted up by 25, we can see that some edges in my face disappear, however other edges detected with min 100 and max 150 still appear.

Facial features appear to be affected the most by changes in parameters in normal lighting. This may be because the lighting is coming from behind me so the contrast in edges on my face are not as extreme as those in well-lit areas. The rest of me, such as my hair or t-shirt, are still well-detected, probably because there is a lot of contrast between black and my skin-tone.

Brighter Lighting from the Front

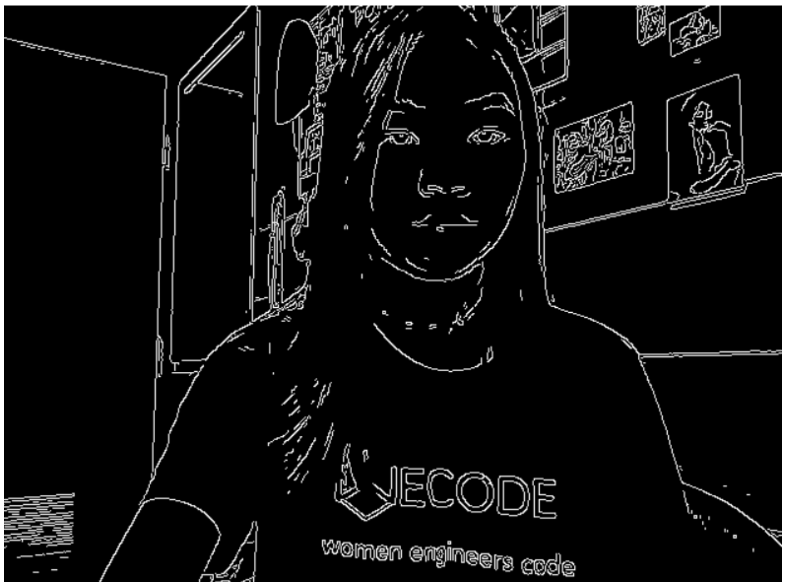


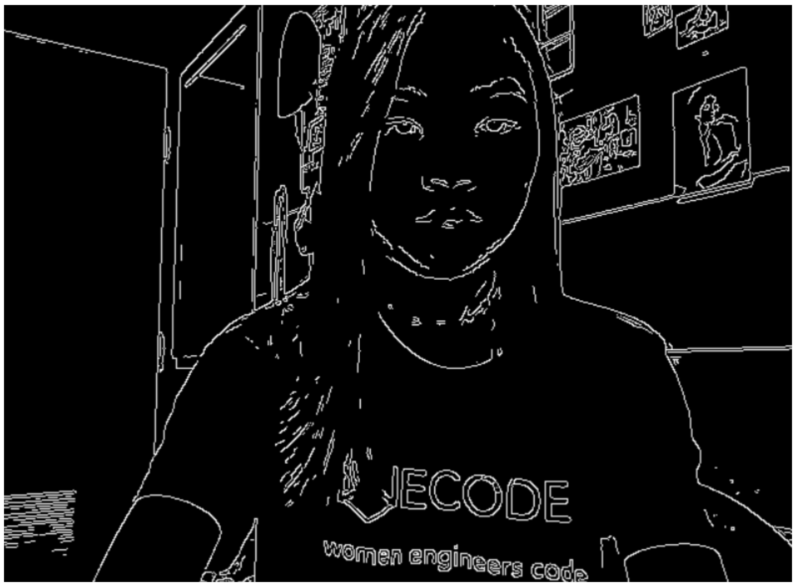


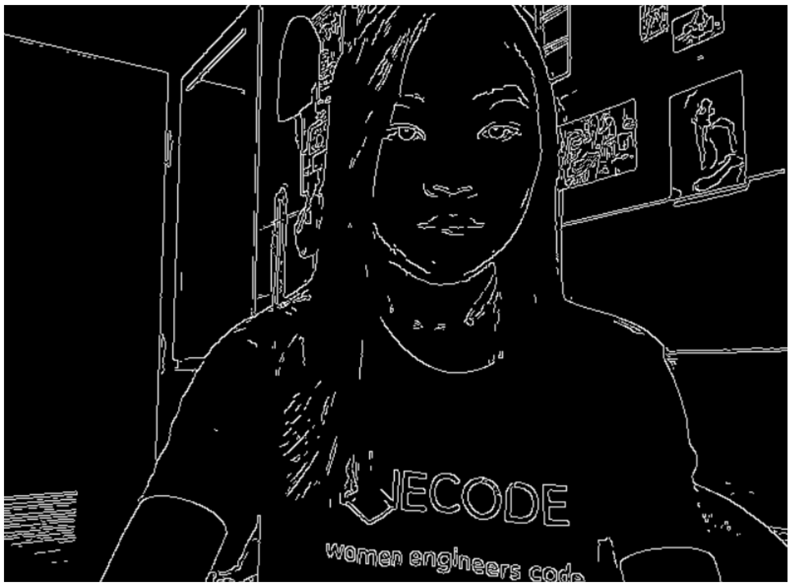


Min 100, Max 150

With better lighting from the front illuminating my face, we can see that suddenly edges in my face that were not detected before become very apparent. With these thresholds, some edges on the sleeve of my shirt were detected. These didn't seem that meaningful, so with a different set of minimum and maximum values, we can avoid this.

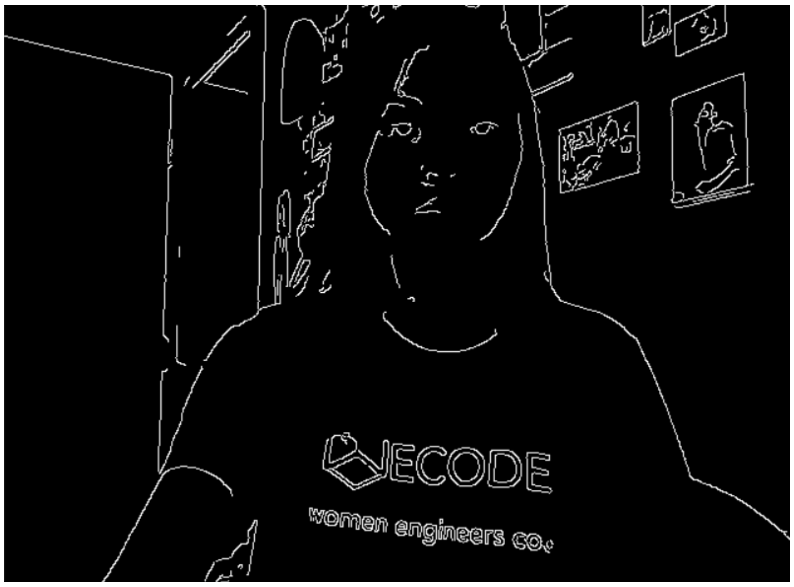
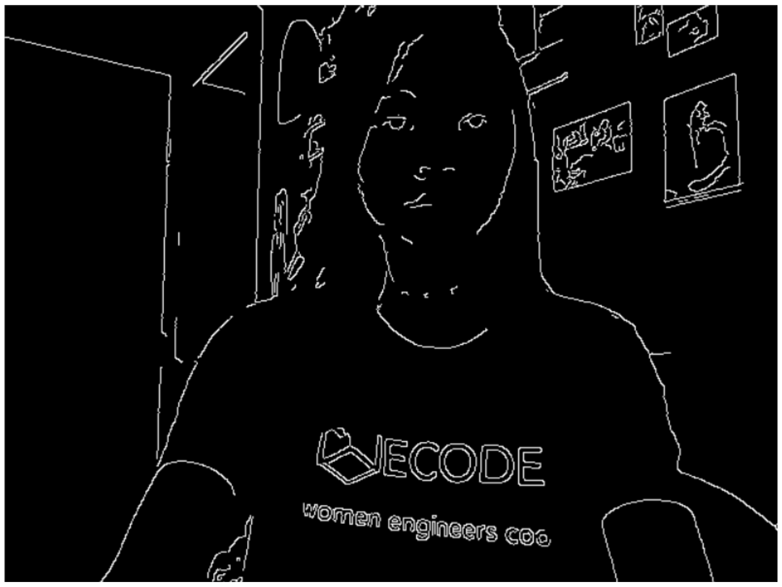

ECODE
women engineers code

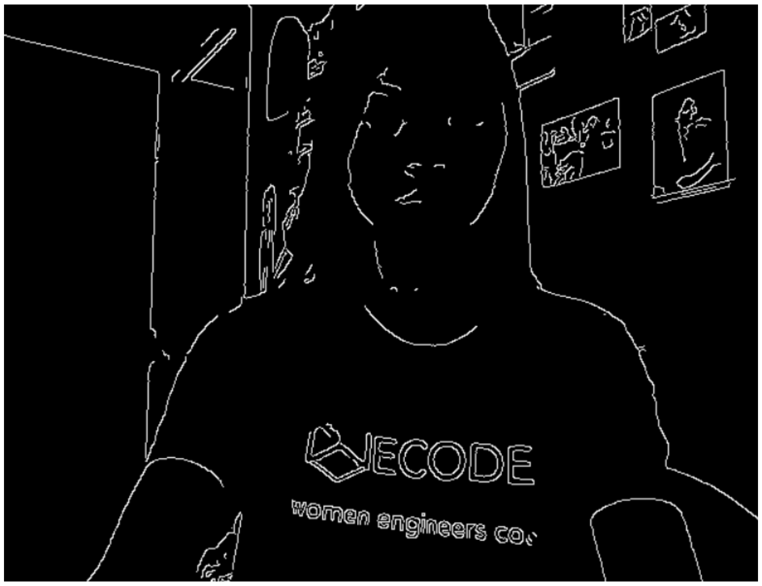

ECODE
women engineers code



Min 125, Max 175

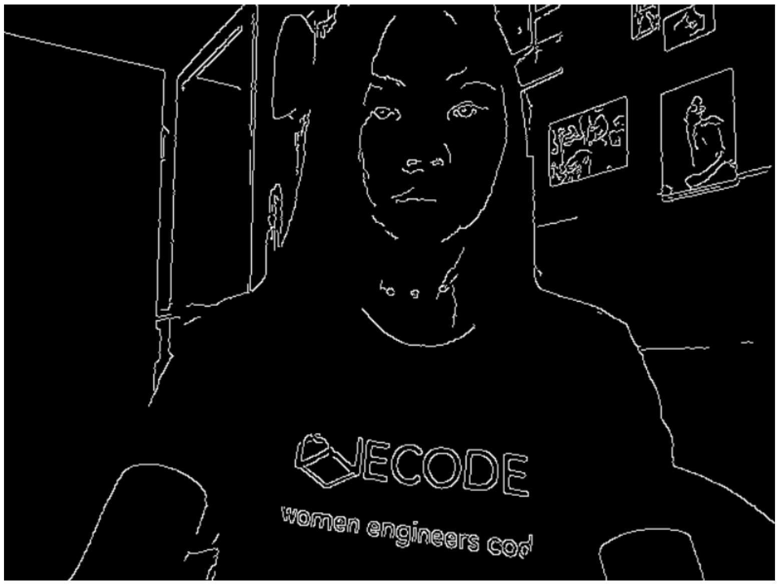
Low Lighting

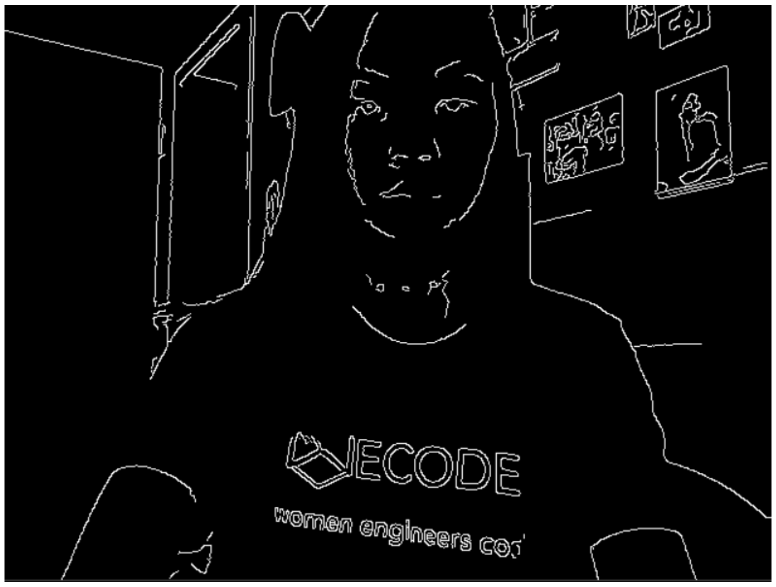

ECODE
women engineers co.

ECODE
women engineers co.

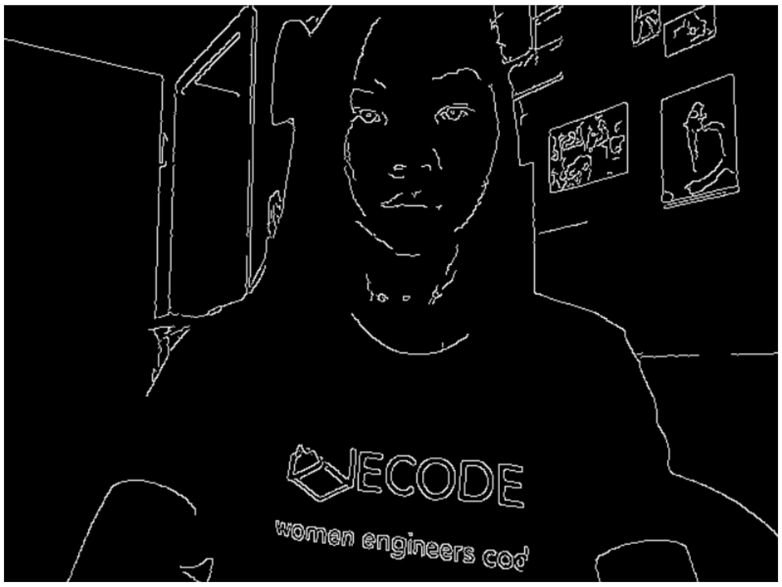


Min 100, Max 150

In low-lighting, we can still make out the objects in this scene, however some meaningful information from my face, the door, and the mirror are lost. Like the issue I had with my facial features in normal lighting, this is probably because in lower lighting, the differences between edges is not as pronounced.







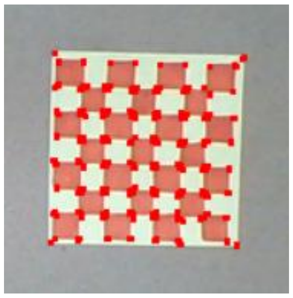
Min 75, Max 125

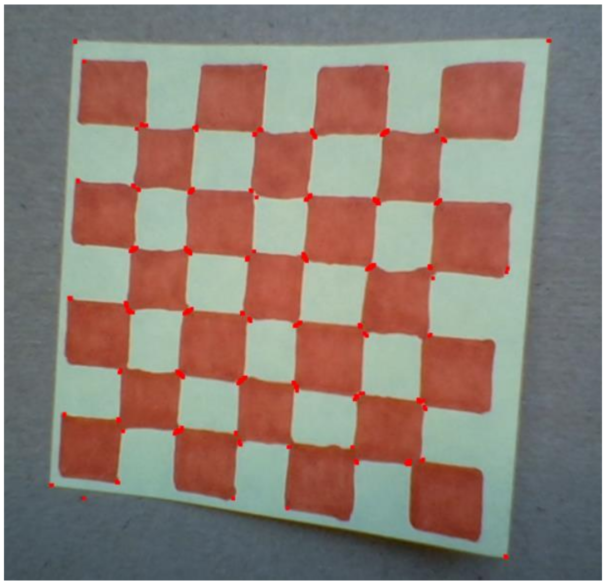
Lowering the parameters by 25 improves the results. We can now see the far edge of the mirror, edges in the walls, as well as facial features. Lowering the maximum threshold probably aided in this, as lower lighting leads to lower contrast which leads to lower intensity gradients. By adjusting our parameters, we can adjust for this change in lighting.

1 Experimenting with corner detection

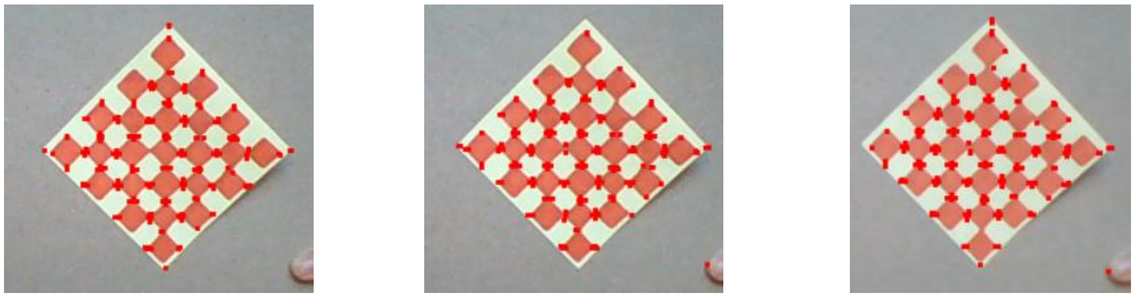
To play around with Harris Corner detection, I made an orange and yellow checkerboard. I decided orange and yellow as opposed to black and white as we might be able to see some limitations with different lighting as orange and yellow in darker lighting would likely lead to less change in intensity than with black and white.

For the below images, I set the size of neighborhood considered for corner detection as 2, the Sobel derivative to be used as 3, and the Harris detector free parameter to be 0.08.


Checkerboard from distance

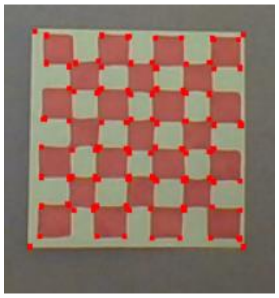

Checkerboard close-up

From a distance, we can see that all corners are picked up aside from one in the lower right. Up close, less corners are detected, which may be because I hand drew this grid so corners may not be as sharp for some of the boxes. However, this also may be a result of the slight angle and slightly lower lighting.

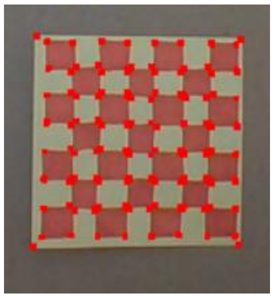


With rotation, we can see that the detector can pick up most corners, but flickers between some.

We see similar problems at an angle, where some corners flicker between frames.

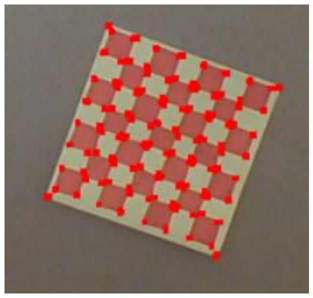


Block size set to 2



Block size set to 3

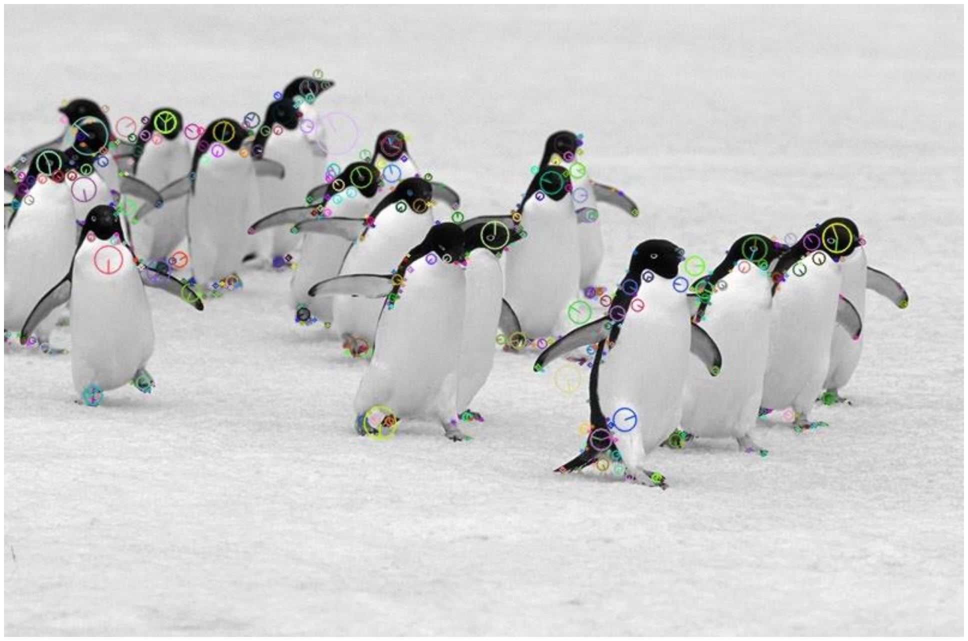
We can see improved results when we increase the block size by 1. Issues with rotations and angles are also improved with this adjustment.



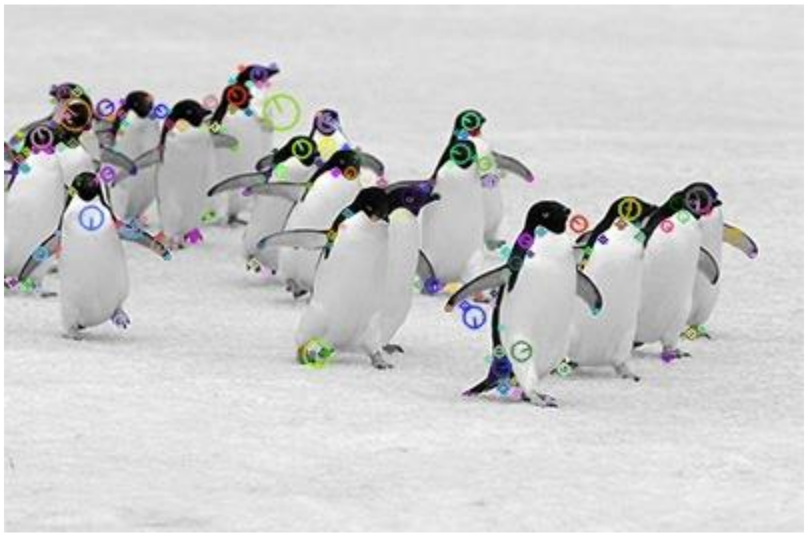
When playing around with the other parameters, performance decreased as less corners were being detected. The detector appeared to only do better with the increase in block size.

2 Experimenting with SIFT descriptors

I decided to play around with an image containing multiple penguins.

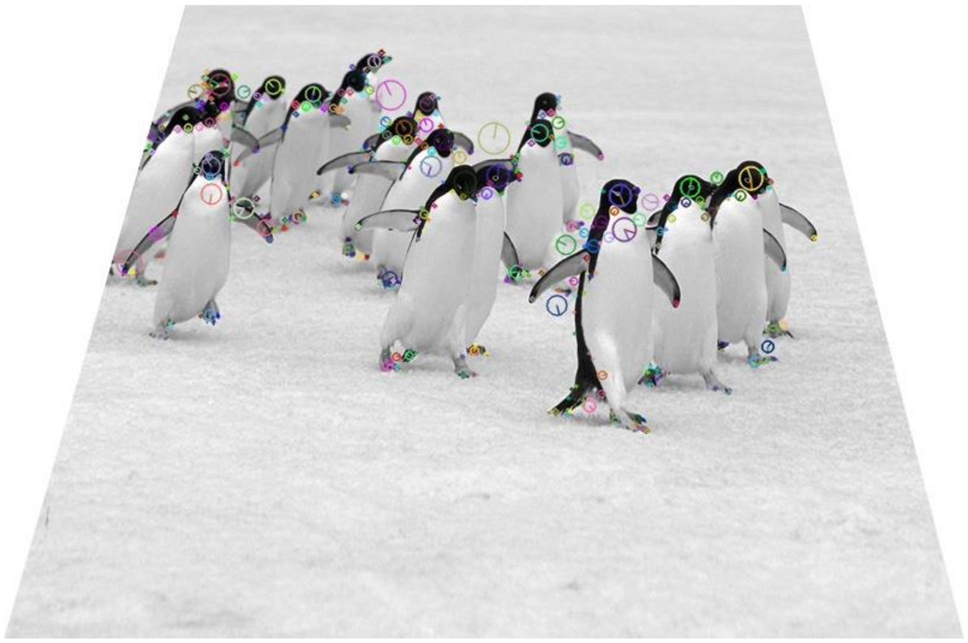


From the default parameters for `cv2.xfeatures2d.SIFT_create()`, I increased the contrast threshold to 10, as before the detector was producing features in the ice that were not meaningful.



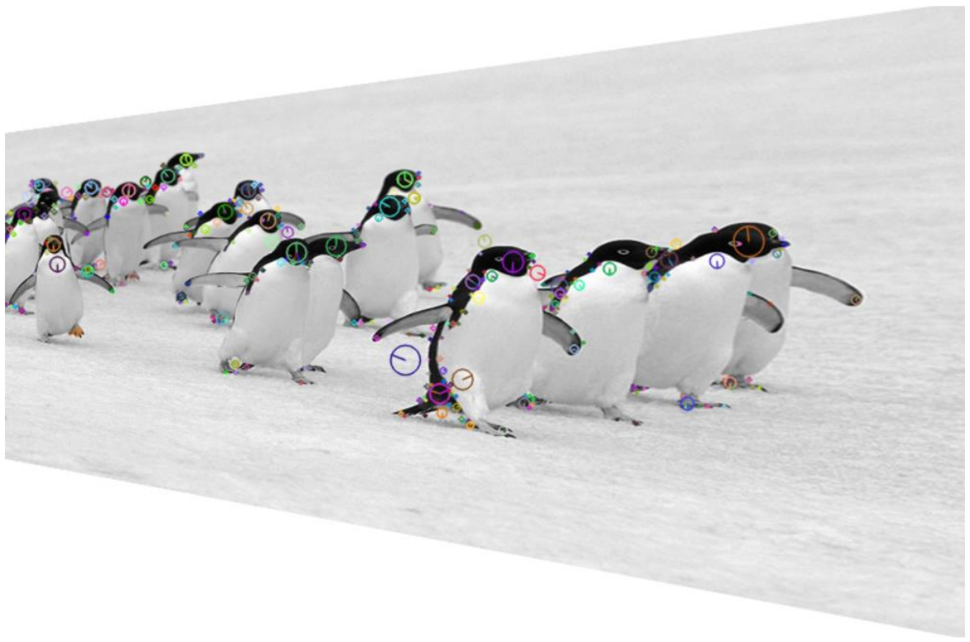
Resized to half of original size

With scale, we can see that most detected features were consistent with the original image. Some features were lost while some were gained. For example, the fifth penguin to the right originally did not have a feature detected on its head in the original image, but in the resized image it did.



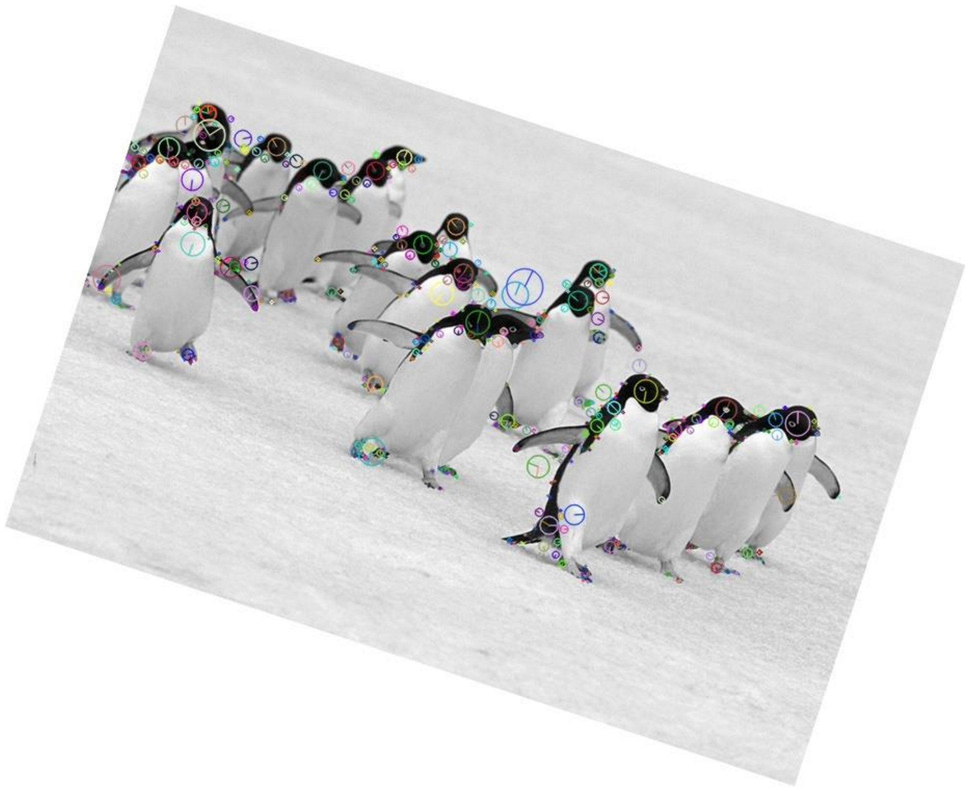
Skewed vertically

With the image skewed vertically, we can see that the detector actually detected more features than in the original image. For example, you can see that a lot of penguins' heads which were not detected in the original image are now detected. This made me curious about the results if we skewed the original image horizontally.



Skewed horizontally

Once again, some features are gained while others are lost, similar to the results in change in scale. It seems that when the penguins' heads are smaller in scale, either from resizing or skewing, the detector is able to detect their heads and other features better.



Rotated

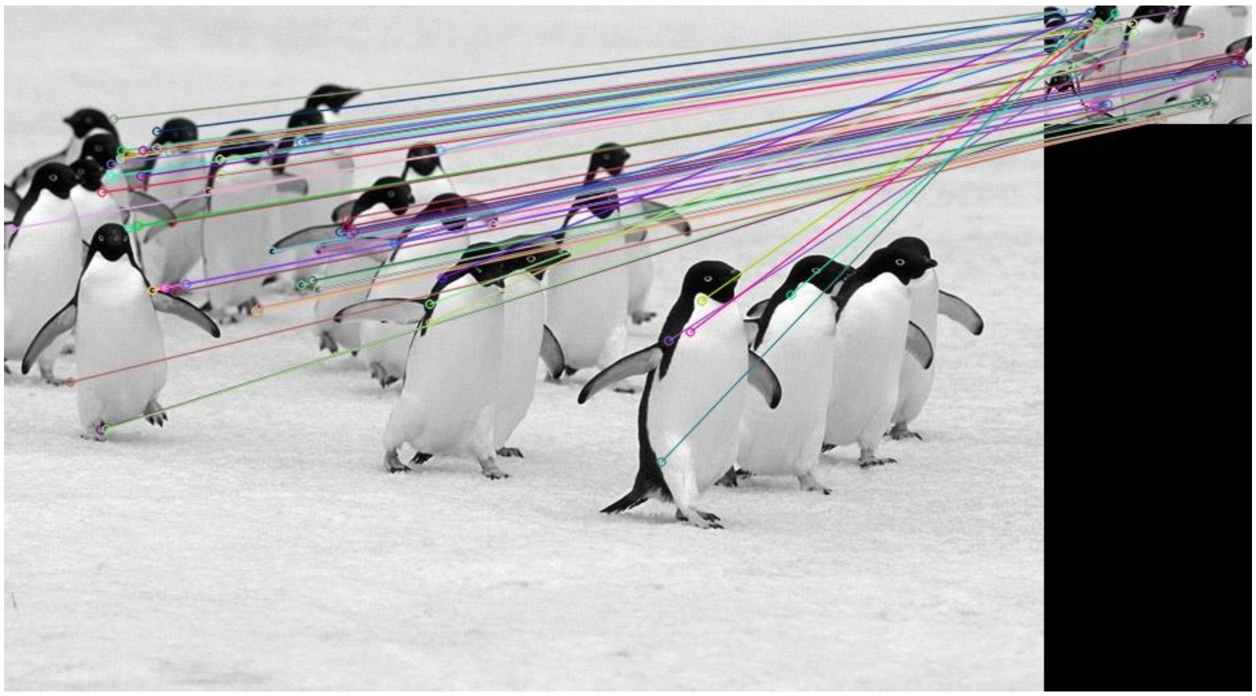
With rotation, we can see a lot more features being detected, both in the penguins and on the ice. Some angles seem to be better than others with the SIFT detector.

Across different conditions, results seem to vary. Under different conditions, however, the detector is able to detect most key features.

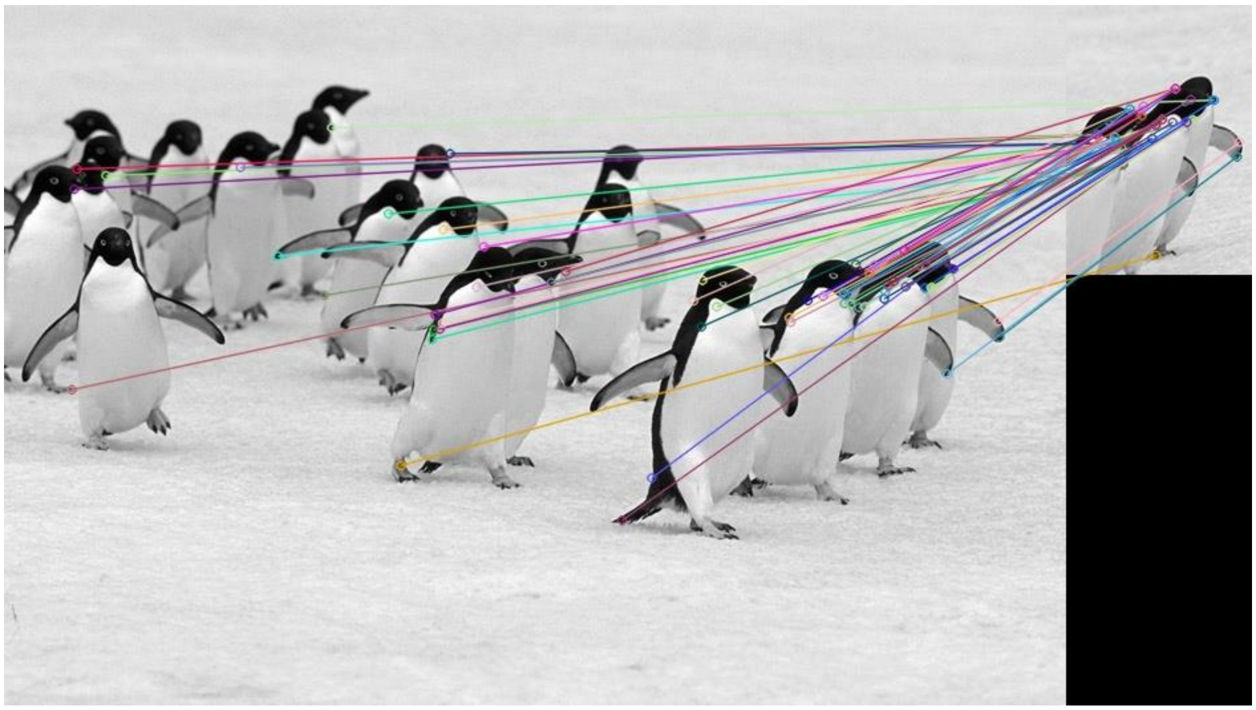
3 Comparing matching performance

For the penguin image, I generated 5 images to match with the original image, that were all manipulated with size, rotation, and skewing.

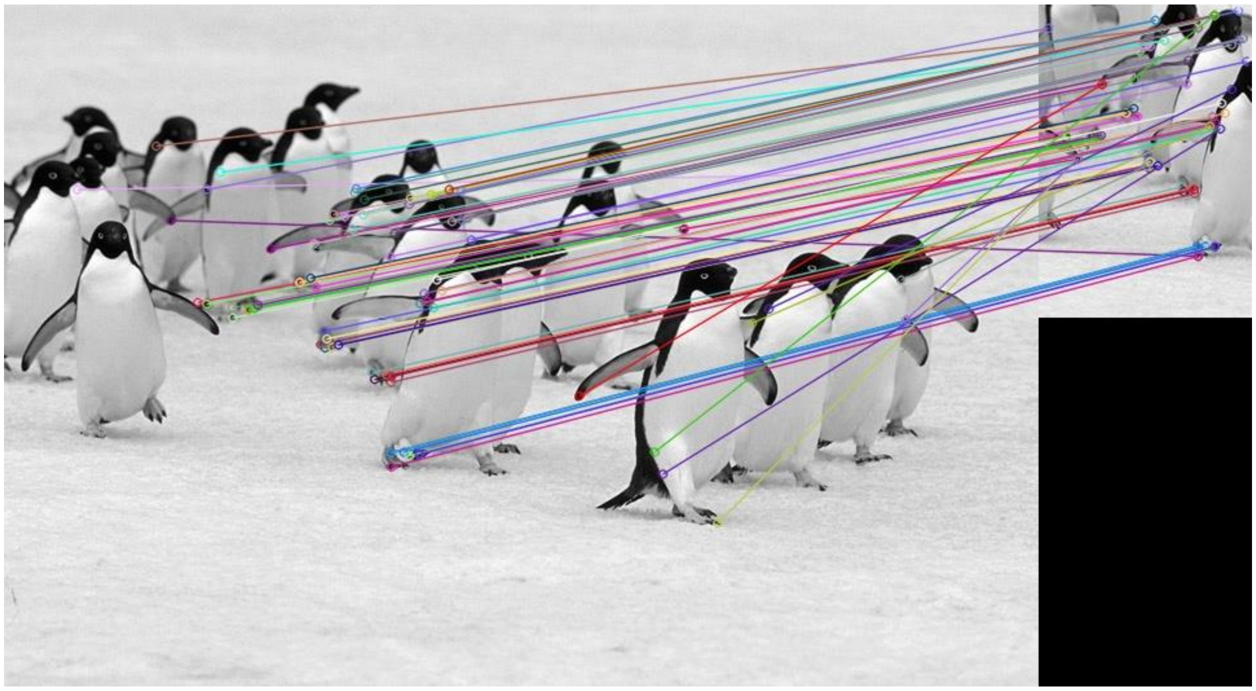
Using keypoints from Harris Corner detection and SIFT descriptors, none of the 5 images I generated had any matches. With just SIFT descriptors, I had more success. The images below were matched using SIFT keypoints.



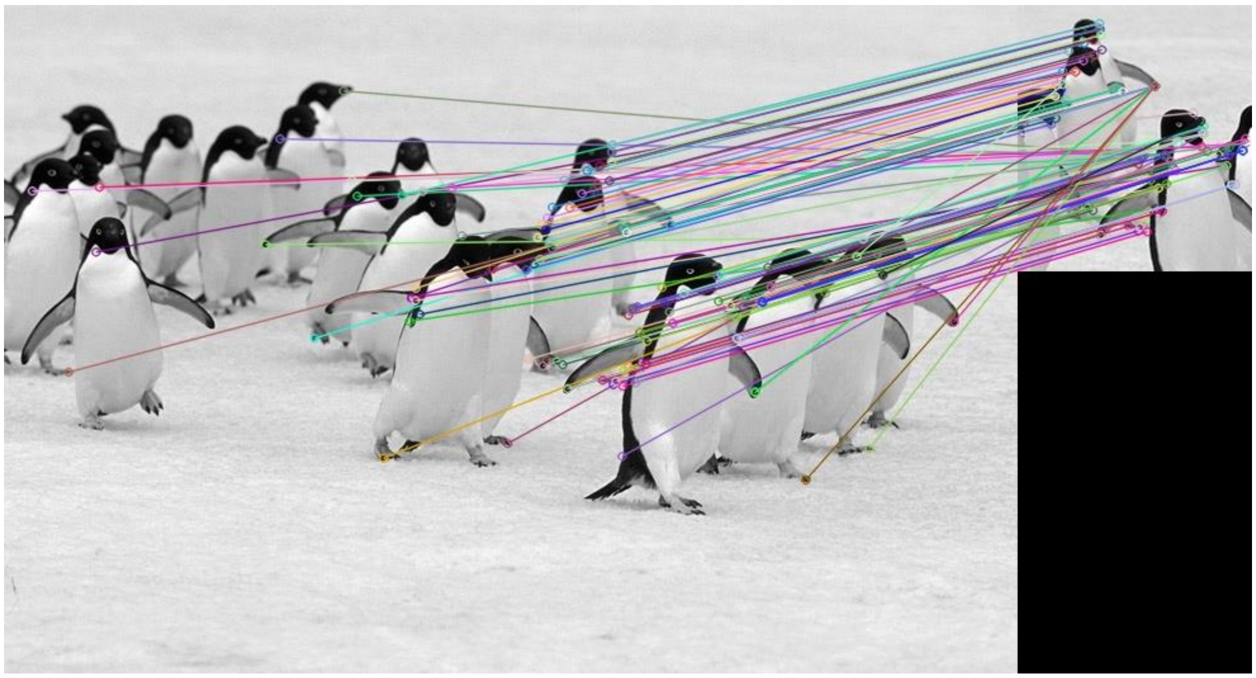
For this image, the majority of keypoints were not matched up correctly. The image we are trying to match was resized, which may be why the keypoints do not match up well.



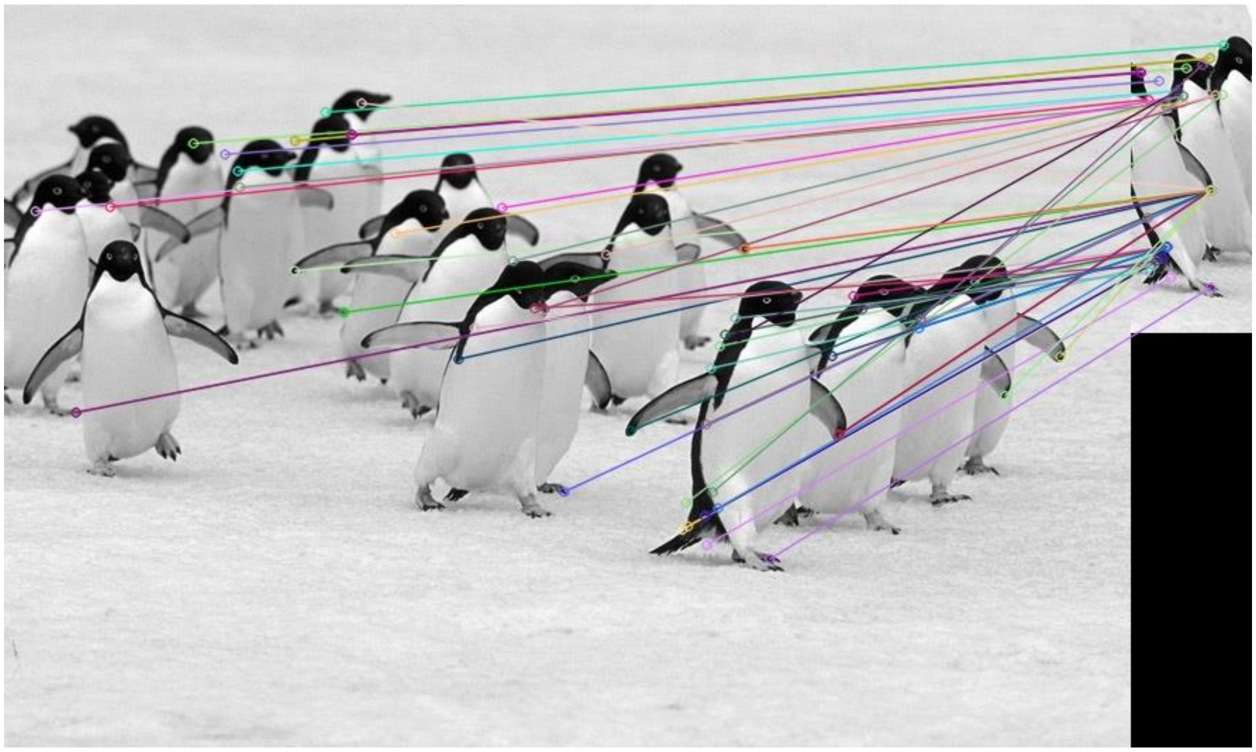
Here we can see a lot of keypoints were matched correctly with the first three penguins, though keypoints were still matched with unrelated penguins.



We can see that a lot of keypoints correctly matched with the original image.



In this image as well, many keypoints are correctly matched.



Here we can see that some keypoints are correctly matched while others are matched with incorrect features. The image we're trying to match with is pretty distorted, which may be the reason.

Feature matching seems to work the best when the image we’re trying to match to the original image is around the same size as the original (hasn’t been resized), and has not been distorted too much, which makes sense. Resizing appears to have a greater effect than rotation and skewing.