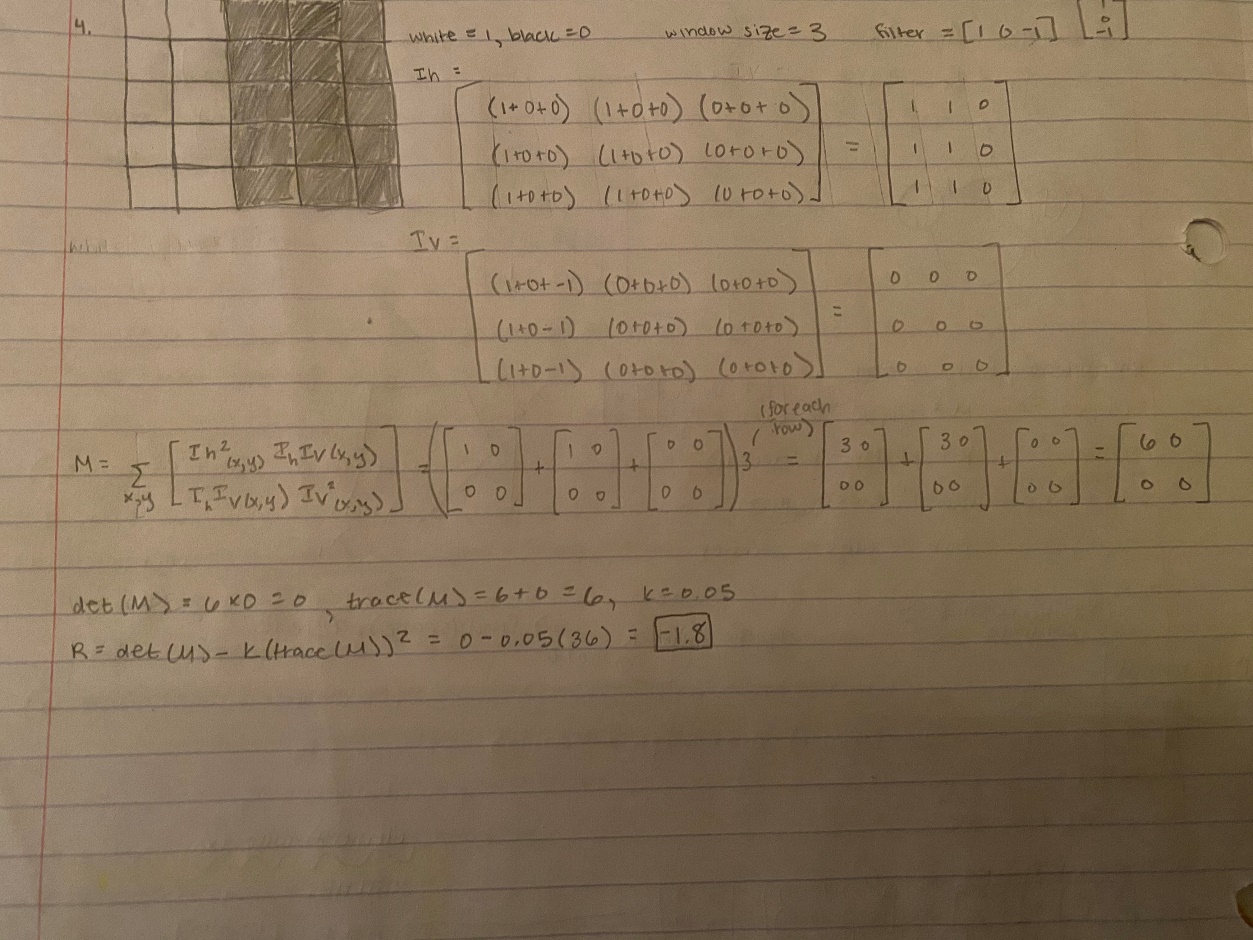
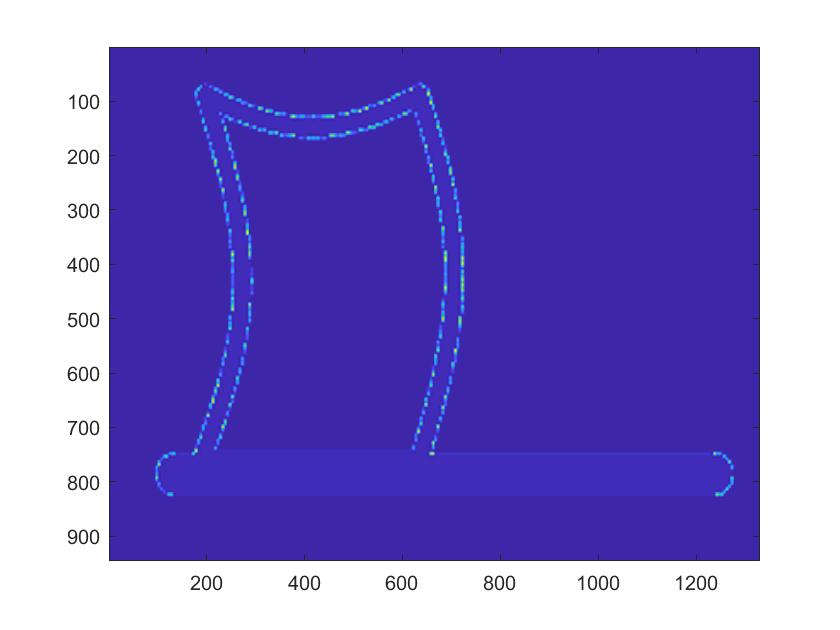
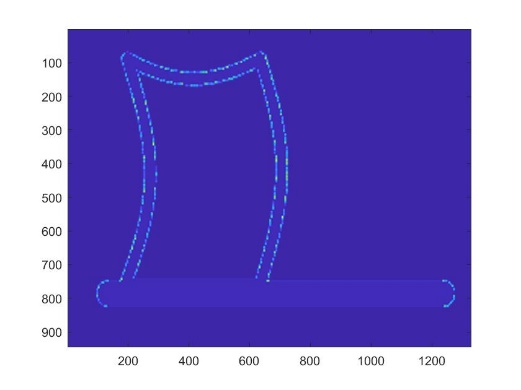
Emma Akbari (eea21) Essay 1

1. One application of computer vision is within robotics; for example, vision in an autonomous robot could allow for movement and other surrounding-based actions. Another application is recognition and interpretation, such as with the conversion of checks to deposit in banking apps. Finally, computer vision can be used for the organization of content. Personally, I find organization to be the most useful in my life. I use applications of computer vision organization on a daily basis. For example, I often search for pictures of a particular category in photos on my phone (anything from a location to a person). I use this similar feature on Google images to find pictures that suit my projects, such as images for a website.
2. If we rotate image A by an arbitrary degree, the sequence of responses would not be the same as if we hadn’t rotated the image. This is because the individual filter orientations are important and not invariant to rotation. For example, if a horizontal edge in A is rotated to become vertical in image B, it will have a high response in the first filter in A but the fourth filter in B. If we compute a mean for both images, they would have a positive distance since their means would be in a different sequence (not invariant to rotation). When calculating Euclidean distance, subtraction between each mean would be nonzero, and thus when squared would be positive. Since this filter bank method is not invariant to rotation, in order to make it invariant, we could instead utilize Gabor filter banks. These filters are based on spectral feature frequency and are both rotation and scale invariant (<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.722.2854&rep=rep1&type=pdf>). Alternatively, we could use SIFT which organizes features by orientation and magnitude; it is rotation invariant via patch rotation using the dominant orientation.
3. One advantage of using filter banks to compute a feature is that it can differentiate between a wide variety of properties. Filter banks, when properly selected, distinguish features based on pattern, orientation, and scale; as a result, two tigers from different viewpoints will yield similar results. Additionally, filter banks can be modified for your intent. For example, if you are writing a program to differentiate between zebras and ladybugs, you can intentionally choose filters featuring stripes for the zebra and spots for the ladybug. Furthermore, filter banks are robust to transformation since gradients stay generally the same with shifts. On the disadvantage side, filter banks may be fooled by noise and interpret an image with its blurred version to be entirely different. Similarly, filter banks will account for illumination changes when the user does not want this (i.e. the same subject at different times of day will be interpreted as different images). Finally, filter banks may not compare images correctly if the user does not carefully select the filters. For example, with the previous zebra/ladybug example, if the only filters chosen identify horizontal and vertical edges, the features will not reveal helpful information.
4. Computing R for center pixel (3,3) [see next page]



1. Corner detection is robust to translation. Within the image patch, the response of a corner will remain the same even if the corner shifts within the image; this is because the gradients will stay the same as the pixels are located in the same arrangement relative to each other. Corner detection is also invariant to rotation on the patch level. The axes measuring the corner rotate with the feature in the patch, so the determinant and trace of M matrix (in Harris detector) remain the same. However, corner detection is not always robust to scale; when an image is enlarged, the corner may appear as an edge instead. In order to make it invariant to scale, you can use automatic scale detection in combination with a function to account for window size changes. Blob detection is robust to translation, rotation, and scale since it takes a local maxima—the features are compared with each other at different levels. Both corner and blob detection are not robust to illumination and noise as they affect the values of pixels.
2. An edge is a collection of points that forms a contouring portion of an image. In other words, it is a line that a human would draw to identify an edge. In order to find edges in an image, we can identify where the largest intensity changes are via the derivative of an intensity function. This would identify the biggest differences between pixels which intuitively is what an edge is represented as. In order to determine the strength of an edge, we can identify the magnitude of the gradient. In order to determine the orientation of an edge, we can look at the direction of the gradient used to calculate it (i.e. a horizontal gradient corresponds to a vertical edge and a vertical gradient corresponds to a horizontal edge). We can use a collection of edges to form a box around objects as a way to identify objects in an image. Each box would compute an “objectness score” based on the number of edges contained solely within that box, and a higher score/number of edges indicates a higher likelihood that an object exists (<https://link.springer.com/chapter/10.1007/978-3-319-10602-1_26>).
3. SIFT is similar to BOW in that they both use histograms for their calculations and are robust to viewpoint changes, translations, and deformations. However, they are fundamentally different in their purposes: SIFT finds descriptors for image features while BOW finds descriptors for images as a whole. SIFT and segmentation via clustering are similar because they categorize parts within an image and they are both fast processes. They are different in that segmentation via clustering is an iterative, back and forth process where points and clusters are continuously updated while SIFT has straightforward calculations. Additionally, segmentation has wider applications since it can group based on intensity, color, and position rather than just gradients. Finally, BOW and segmentation via clustering are similar because they could be sensitive to illumination changes. The two techniques differ in the same way that SIFT and BOW differ: BOW is a descriptor of the image as a whole while segmentation focuses on the clusters within a single image.
4. For this example, I took a flag image and applied scaling and rotation. The scaling matrix = [5 0;0 5], so both the x and y coordinates are multiplied by a scale of 5. The rotation is by 90 degrees. Therefore, the matrix = [cos90 -sin90; sin90 cos90] = [0 -1; 1 0]. The final output of the flag in the scale then rotate image was the same as the output of the rotate then scale image. This makes sense since scaling and rotation are independent actions that do not impact one another.
   1. scale then rotate
   2. rotate then scale
5. One example of dropping information is linking and thresholding within a canny edge detector: the higher threshold gets rid of more edges, ending with a simpler final image of edges. A pro of dropping this information is that extraneous “edges” will be discarded (such as from noise) while a con is that some edges may be lost if the chosen threshold is too high. Another example is smoothing via a Gaussian filter; only the low frequency components pass this filter. The Gaussian filter can help keep only the most important information while reducing noise, but it does not preserve image edges since it uses a mean. In texture representation, we can also discard all information other than that provided by gradients (aka describe textures using only gradients such as horizontal/vertical). By focusing solely on gradient detected texture, we can easily differentiate images (most will not have the same exact gradients), but the images may not be robust to geometric changes since it will change the gradient calculations. A fourth example of discarding information is reducing image size by subsampling. This can create a smaller image in a simple way, but it may warp the image since the most important aspects are not always preserved (for example, it may create an unwanted ripple effect). Finally, information is discarded in non-maximum suppression during the Harris detector algorithm. This creates a more reasonable number of key point features in an image, but it can suppress too much information if many key points have similar R values (the selected key points would then be highly dependent on the implementation of suppression). Overall, this process of information dropping reminds me of the human tendency to use heuristics. Upon analyzing novel images, we focus primarily on aspects that are familiar or standout, giving us a limited interpretation (based on previous knowledge from Psych classes).
6. I would want to help build a computer vision system to watch the environment to identify and notify us of crisis’s. This could entail a few different elements of our changing environment: it could monitor sea levels, ice cap melting, and significant decreases in wildlife. This would be difficult to implement since the machines watching the environment would need to be strategically placed. Engineering teams would need to decide whether satellite, drone, or land machines would be best and the prices for these devices would be expensive. Additionally, in order to design algorithms that identify and alert us of problems, we would probably need years of previous data as well as training and testing at each location in order to identify significant events (e.g. how much ice melted is worrisome? How do we account for the distance/angle of the monitoring machine in calculating ice amounts?). Potentially, this could cause problems if incorrectly implemented by false positives or false negatives, aka alerting us when there is not a crisis and not alerting us when there is a crisis. However, I think a computer vision system to monitor the state of our environment could help us to address climate change.