Gordon Lu

CS 1674

Dr.Hwang

Essay 1

1) When I walk down the streets of Pittsburgh, sometimes I'll see Uber's self-driving cars. Self-driving cars, or broadly speaking, robotics require computer vision for perception so it can classify and detect objects so it can avoid colliding into pedestrians and other things such as mailboxes. Social media in the 21^{st} century occupies a lot of our days, here, we will find another application of computer vision in the form of Snapchat filters. Putting dog ears, or flowers around your face requires computer vision, as it uses image processing and face detection to determine how big to make the flowers or dog ears. It's also why if we try to apply a Snapchat filter on something not human-like, it will struggle and more often than not fail to put the flowers or dog ears. An application that is more mundane and academic related is scanning apps. Broadly speaking, for organizational purposes. Any time I have to scan a document using a phone app, this requires computer vision. Computer vision is relevant here as the camera needs to be able to detect the paper, as well as the handwriting. Thus, scanning requires object detection. I would consider a scanning app to be the most useful, broadly speaking, any sort of app to detect objects and analyzing the contents, such as bank apps to deposit checks are very important. With COVID-19 still in the air, the necessity to avoid contact has never been at an all time high. The ability to just scan a check and deposit the money right away rather than drive to the bank is a huge positive and relieves what would be a mundane task. Also, with apps that allow me to scan work, it reduces the stress of having to type work up after I have completed it or hand the work in directly. As someone who always prefers things to be done in the easiest way possible, scanning apps are the most appealing and beneficial, as it makes mundane tasks trivial.

2) If Image A is rotated by an arbitrary degree, the sequence of responses to the filters would not be the same as if the image was not rotated. Each of the individual filter orientations are cruicial to the rotated image, B, and thus are not invariant to rotation. Consider if a vertical edge in the original Image A is rotated such that it is now a horizontal edge in Image B. This newly rotated edge will have a high response in the fourth filter in Image A, but a high response in the seventh filter in Image B. In terms of

distance, the L2 norm or Euclidean distance between each of the means would be non-zero as the images are not the same. Since the distances are non-zero, when the L2 norm is calculated, the results would be positive. Since the filter bank described in the question is not invariant to rotation, an alternative descriptor would be a Tunable Gabor Filter, which is tuned according to scale and rotation changes using polar coordinates of a frequency spectrum. Gaussian Tapering is then applied to properly reduce spectrum leakage due to discontinuities at end points of a waveform. A paper written by Xinqi Chu and Kap Luk Chan was able to conclude that using Tunable Gabor Filter Banks achieved concurrent rotation and scale invariance 44% better than typical Gabor Filter Banks (http://www.ifp.illinois.edu/~chu36/research/rsi_texture.pdf). Alternatively, SIFT could be used, in which features are organized according to orientation and magnitude. SIFT is rotation invariant through patch rotation using the dominant orientation.

3) Filter banks have many advantages. One of which is the ability to compute a feature so that it can deifferentiate between a variety of properties. When properly selected, filter banks allow us to distinguish features based on pattern, orientation and scale. As a result, an object like an apple at different views will yield similar results. Filter banks can also be tuned based on intent. For example, imagine I want to write a program to determine the differences between a cheetah and a tiger. A filter bank can be chosedn to feature spots for the cheetah and stripes for the tiger. Filter banks are also robust to transformations, as the gradients tend to stay the same with shifts. Where filter banks start to falter is with noise. Similar to what was done in homework 2, when images where blurred, it became difficult to determine what was a panda, and what was a cardinal. With filter banks, if the filters are chosen erroneously, the images may not be compared correctly. Often times, filter banks will also account for changes in lighting, even when the user may not want this. In the example of the cheetah and the tiger, if a filter is chosen to identify horizontal and vertical edges, the resulting features will not help much. A similar case can be drawn between the cardinal and the panda, if a filter focuses on what the panda is eating or what the cardinal is standing on, it will not help as much as identifying something like spots on the panda or the beak on the cardinal.

4) Corner detection is robust to rotation on the patch level. The axes measuring the corner rotate with the feature in the path, so the eigenvalues (shape) reamins the same. Corner detection is also robust to translation on the patch level. The response of a corner on the patch level be the same, as the window function and the derivatives are shift invariant. So, the gradients will remain the same, since the locations of the pixels are located in the same arrangement relative to each other. However, corner detection is not robust to scale. When an image is zoomed in for example, we cannot use the same window size or the same threshold, as corners may be misclassified. In fact, most corners become less "sharp" when zoomed in, and may not be detected as a corner. To solve this problem, we would use automatic scale detection in tandem with a function that accounts for the same shape, regardless resizing. With blob detection, or the Laplacian of Gaussian, it is robust to rotation, translation and scale, as we find a local maxima in position-scale space. In other weords, the features are compared to each other at different levels. However, both corner detection and blob detection are not robust to illumination and noise, as said transformations impact the values of the pixels themselves.

5) An edge is a place of rapid change in the image intensity function. In other words, edges would correspond to the extrema of the derivative of the internsitve function. It is a line that a human would draw to identify an edge. To determine where an edge lives in an image, formally, we would use an edge detector like the Canny edge detector. Generally, we would filter a given image with the derivative of a Gaussian, find the magnitude and orientation of the gradient, and identify where the largest changes in intenstive are using the derivative of an intensity function. This would allow us to identify where the biggest difference between pixels are. Intuitively, the location where the largest difference in pixels is what an edge is represented as. To determine how strong an edge is, we can identify the mafnitude of the gradient and to determine the direction of the orientation, we would find the orientation/direction of the gradient used to calcualte it. In particular, a horizontal gradient would correspond to a vertical edge, vice versa. Using a collection of edges, we can form circles using the Hough transform. This is done by transforming every edge to all possible lines that could pass through said point, and map this to polar coordinates to form circles. In Hough space, the possible centers of circles containing all edge points (x_i, y_i) will be computed and the center that has the most intersection will be considered the detected circle. Another complex structure that can be formed using a collection of edges is a box. This box would be used to identify an objects within an image, similar to

how in homework 5, we used the Hough transform to detect circles in an image, we can instead form a box around objects to identify an image. Each box would compute an "objectness score" based on the number of edges contained within said box and would classify an object based on some threshold and accuracy to determine whether an object exists. Additionally, a higher "objectness score" and number of edges would indicate a higher likelihood that an object exists (https://link.springer.com/chapter/10.1007/978-3-319-10602-1_26).

6) A SIFT representation for a key point and retrieval or classification based on a BOW representation are similar in the sense that they both involve the computation of a histogram to find the most dominant gradient orientation (SIFT) or to summarize an image based on word occurrences. Both SIFT and BOW are robust to viewpoint changes, translations, and deformations. However, where SIFT and BOW differ are in their purpose. SIFT and BOW are fundamentally different. SIFT finds descriptors for image features, whereas BOW finds descriptors for images as a whole. With SIFT and segmentation via clustering, both aim to categorize parts of an image, and both are respectively fast. Segmentation via clustering is an iterative, back and forth process in which points and clusters are continuously updated. SIFT, on the hand, is calculated through one pass, and is straightforward. Segmentation also has a wider range of applications, as pixels can be grouped based on a feature space, which could be color, intensity, etc., rather than being limited to what SIFT does with just using gradients. With BOW and segmentation via clustering, both are sensitive to illumination changes. However, BOW and segmentation via clustering differ in the same way that SIFT differs from BOW. BOW finds descriptors for images as a whole, while segmentation via clustering will focus on the clusters within a single image.

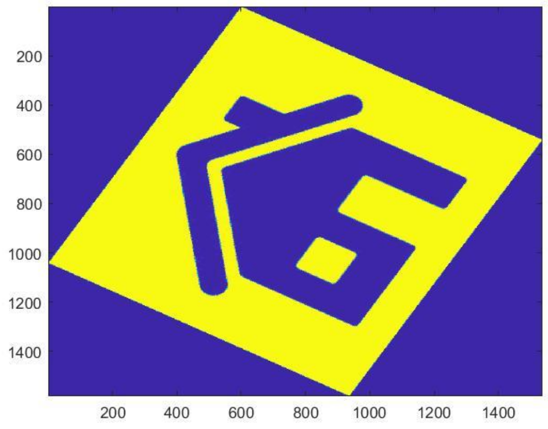
7) I considered a simple picture of a house, which was the following:

I applied two simple geometric transformation, one being a rotation and the other being scaling. I considered scaling the image by 3, so the corresponding scaling matrix was [3 0;

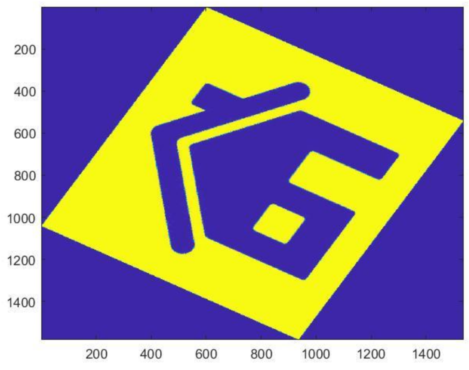
0 3], so that both the x and the y coordinated were multiplied by 3. As for the rotation, I considered a 60 degree rotation. The corresponding rotation matrix was:

$$\begin{bmatrix} \cos(60) & -\sin(60) \\ \sin(60) & \cos(60) \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix}$$

As a result, by first scaling the image, then rotating the image by 60 degrees, the result was the original image scaled by 3, rotated counterclockwise by 60 degrees, which was the following:



Similarly, performing the rotation by 60 degrees first then scaling the image by 3, resulted in the following:



Comparing the two final outputs, we can clearly see that they resulted in the same image. This is logically sound, as scaling and rotation are geometric transformations that are independent of each other.

8) A common example of dropping information involves linking and thresholding with a canny edge detector. Higher thresholds will get rid of more edges, which results in a simpler final image of the edges. One positive of dropping said information is that erroneous/extraneous edges will be discarded, these edges could include things that could result from noise. On the other hand, a negative is that some important edges may be lost is the chosen threshold is too high. Another common example of dropping information is smoothing through a Gaussian filter. Through this Gaussian filter, only low frequency components will be able to pass through. It can help keep only the most relevant/important information, while reducing noise, however the Gaussian filter does not preserve the image edges, as it is computed using a mean. Texture representation, similar to a Gaussian filter will also discard certain information. However, in the case of texture representation, only information provided by the gradients. A positive of this approach is that by focusing only on gradient detected textures, we can easily distinguish images, as most images will likely not have the same exact gradients. One con is that the images may not be robust to geometric transformations, as said transformations will change the computation of the gradients. Another example of discarding information is with reducing image size by subsampling. Subsampling can produce smaller images in a simple way, but it may also warp the images, as the most important features/aspects are not always preserved. A fifth example of dropping information comes in the form of non-maximum suppression in the Harris detector algorithm. Non-maximum suppression creates a more reasonable number of key point features in an image. One con of this approach is that it can suppress too much information is many key points have similar R values. The entire idea of dropping information reminds me of efficient studying habits. In a world where time is not a sparse resource, to truly study well and retain all the information in every little slide and word in a textbook would not be feasible, especially when studying for an exam. Dropping reminds me of when I employ efficient studying habits, and rather than look at every single small detail in slides and materials to study, I look at the key points, the highlighted words, I scan and focus on big words, rather than getting stuck on the little details. Dropping also reminds me of efficient techniques to perform well on the SAT. Given a large corpus of text, the SAT imposes a time constraint on how long students have to analyze text and make inference from said text. Of course, with enough time, a lot of students would be able to get the answers to the inferential questions. However, with limited time, we are often taught to look for the key points, rather than looking at tiny details. We don’t look at every character, but we look at the big picture, and what matters in the grand scheme of things.

9) Ideally, I would want to design a computer vision system that would be able to make inferential decisions quickly. In particular, something along the lines of monitoring stocks, and being able to make decisions to buy or sell based on a graph of the history of a stock. The computer vision system would be presented the graph of the stock, and based on the trends it sees on the graph, it would infer what decision to do with said stock. The complexity of this system would be beyond belief. Not only would the computer vision system need to be able to recognize a graph, but it would need to be able to tell when something is bad and good. This system would replace the tasks of algorithmic traders, as well as statisticians, as the idea of being able to look at a graph and generate some text based on the interpretation of a graph is what many statisticians and data scientists are hired to do. The benefits of such a system are infinitely large, not only would it allow for a reduction in the amount of time to analyze and make inference for a human, it has the potential to save a company hundreds of thousands of dollars to hire quantitative analysts. A clear problem that this system could cause is that it can be millions of people into unemployment. One of the most attractive things about machine learning and data science is the large salary it can bring. With a dominant part of what machine learning, data science and statistics done in the form of a system that can automatically generate very accurate inferences based on an image, there would be quite a bit of unemployment. Another problem is the amount of money required to research and fund such an ambitious project. If such a project were to be able to be funded, it would cost trillions to fund, as the cost of being able to create a system with such accuracy would take years and many large data sets to test and train to find optimal hyperparameters to train the data on. A lot of knowledge required to be able to build such a system would fall into the realm of machine learning, recognition, text generation, and detection. On one hand, once we have data, what machine learning model to use, such as using neural networks, and how to use gradient-based optimization algorithms, and methods such as cross validation to find the best model would require more knowledge. Beyond ideas like KNN, CNNs, Cross Validation, there are certainly more advanced machine learning models that suit computer vision, as well as text generation well. On the hand of text generation, how can the system generate text, or come up with some way of writing good analyses based on a graph? Also, how would the system be able to detect the graph and know if the stock is doing well or not based on said graph. Additionally, how would the system be able to detect if small creases/kinks are not points in the graph? The implications of what this computer vision system would be able to accomplish by being able to produce accurate analyses like a human would is vast. Not only would it be able to replace a lot of jobs in the world, but it would allow for less wasted time to focus on more pressing manners and accelerate a lot of tasks in the workplace. The main drawback is the time, resources, and the huge gaps in knowledge to create such a system.