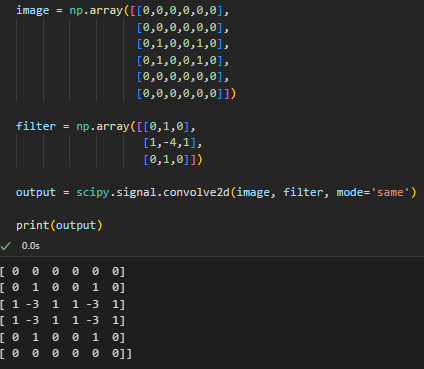
sglan1.a.i) The output matrix is the same size as the input, so we must have some sort of padding.



1.a.ii) This is a Laplacian of Gaussian Filter. It computes the summation of the second derivative in the x direction and the second derivative with respect to y. It is useful in computer vision when calculating interest points. the LoG filter is a versatile tool for edge detection, feature extraction, and blob detection in computer vision, and it can be used in a wide range of applications, including object recognition, image segmentation, and image processing.

1.a.iii) This filter gives heavy negative weights to points of interest. We could instead flip the sign of each element, which would give us a positive response at points of interest? Another disadvantage of the LoG filter is that it can produce a large number of false positives and false negatives in certain situations. For example, if the image contains regions with high frequency noise or sharp transitions, the LoG filter can produce spurious edges features that do not correspond to real structures in the image.

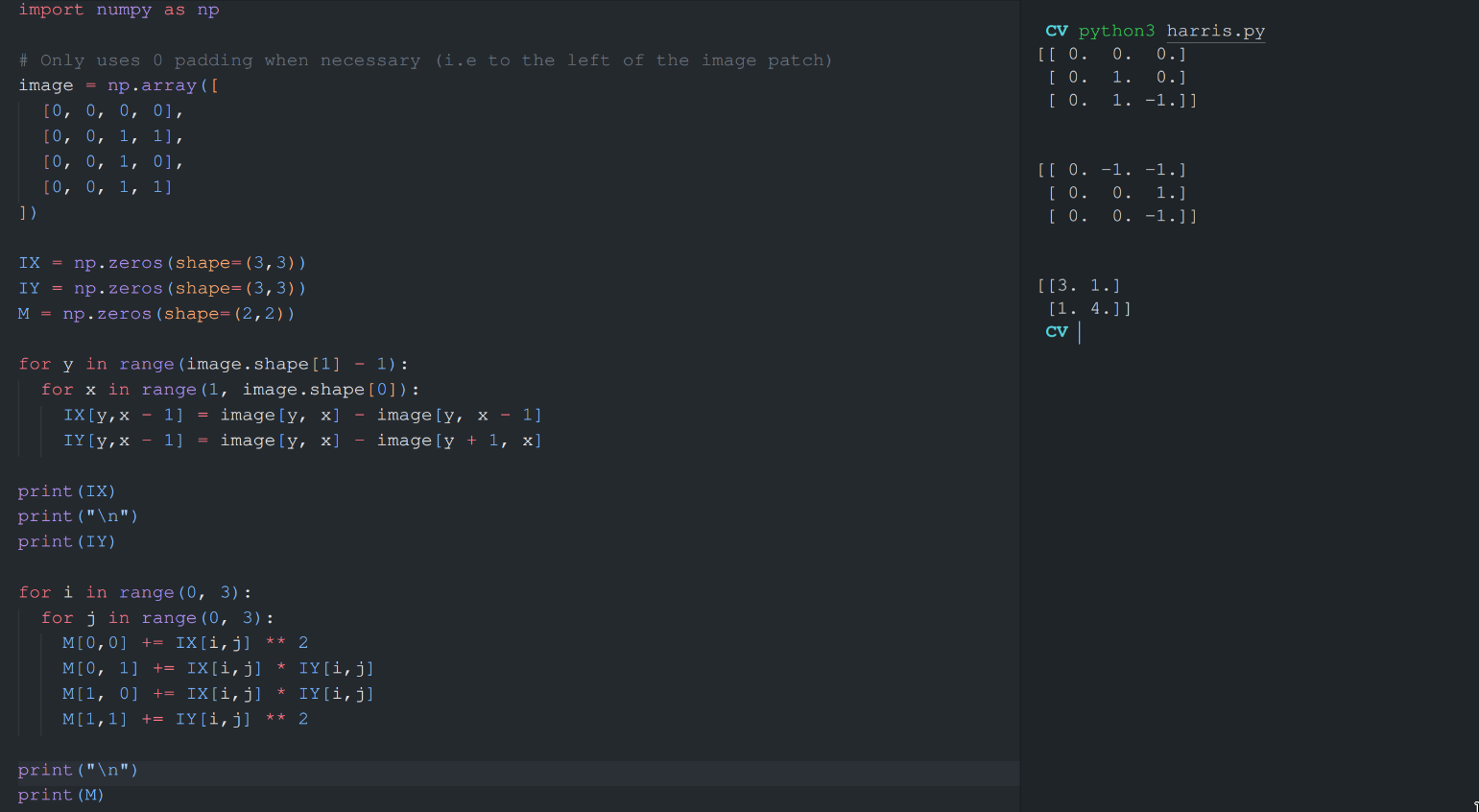
We can alleviate this issue by combining this filter (convolving) it with a moving average like a gaussian filter which would filter out high frequency noise and give a better and more accurate response to blobs / points of interest.

1.b)

## 

## Cryptography

Harris Detector Response,



If you don’t use zero padding below the image patch then I believe the result comes out differently. Can someone validate whether we should use padding below the image patch despite not requiring it?

2.a.i)

Interest points in the SIFT algorithm are detected by using the Difference of Gaussian at different scales. That is, for each we choose, ~~we calculate the gradients using either the Sobel filter or the Prewitt filter in both the x and y directions, then for every pixel~~ we calculate .

Therefore, the light areas surrounded by dark objects will receive a strong negative signal because the gaussian smoothing of a larger filter will darken the center, therefore, a small value – a large value will give a negative response. The same opposite argument holds for dark spots on light backgrounds.

We then localize interest points detected by the DoG to find the refined estimates

2.a.ii)

* ew

2.b.i)

In order to estimate the transformation parameters (𝑠, 𝑅, 𝑇) between two images related by the given transformation, we need a minimum of three corresponding interest point pairs between the two images.

This is because the transformation has three degrees of freedom: two for the rotation matrix 𝑅 (since it is a 2D rotation), one for the scaling factor 𝑠, and two for the translation vector 𝑇. Each corresponding interest point pair provides two constraints (one for each image), so we need at least three pairs to fully constrain the transformation

2.b.ii)

[Any ideas?]

Probability of an individual iteration being successful (at least three of the five points are inliers) ((brackets are a big assumption, might be wrong, don’t judge me)):

0.1^5\*0.9^0\*5C0 + 0.1^4\*0.9^1\*5C1 + 0.1^3\*0.9^2\*5C2 = 0.00856

Probability a good fit is not found after n iterations:

(1-0.00856)^n = 0.99144^n

Alternative: probability of getting 5 inliers = (1/10)^5. So the probability of not getting this after n iterations is (1 - (1/10)^5)^n.

3.a.i) Just look at slides, essentially just a histogram for each section of the image with overlap and normalized.

3.a.ii)

Because we don’t have any training labels this is an unsupervised learning task. Our task is to categorize the images into different classes:

The images are not ‘ripe’ for processing because there is a lot of Ambient light in a lot of these images, some are in plain white backgrounds, and some are straight-up in a forest we need to perform some pre-processing that would attempt to normalise the images so that we know which area to focus on. Perhaps, even turn them into black and white, since these plants don’t seem to have vibrant colours. There are also pictures with multiple shapes, we can segment the image into more tiny ones so we can focus on one image at a time.  
Furthermore, each image is of a different shape and dimension, so in order to not confuse our model into learning features that are totally not related to the plant, and related to the shape and size of the plant in the input we should try to crop out the image into a pre-determined image, but such an image that is good enough resolution to fit all kinds of plants in.  
Because there are going to be a lot of similarities between classes, we don’t want a hard cut-off point between plants, so I propose we use gaussian mixture models, so that there is a smooth probability of any plant of belonging to any class.  
~~To play things safe, each class in the feature vector will have a probability of being in that class. We can perform a softmax function to more easily identify the most likely candidate.~~

Alternative: K-means clustering of the HOG descriptors (which are already normalized).

3.a.iii)

Perform identical pre-processing as for the stop mentioned above and insert into the model. The model will return a vector of probabilities of that input of being in each class. We can select the largest set of probabilities as being the most likely to belong to that class.

（Or perhaps KNN?)

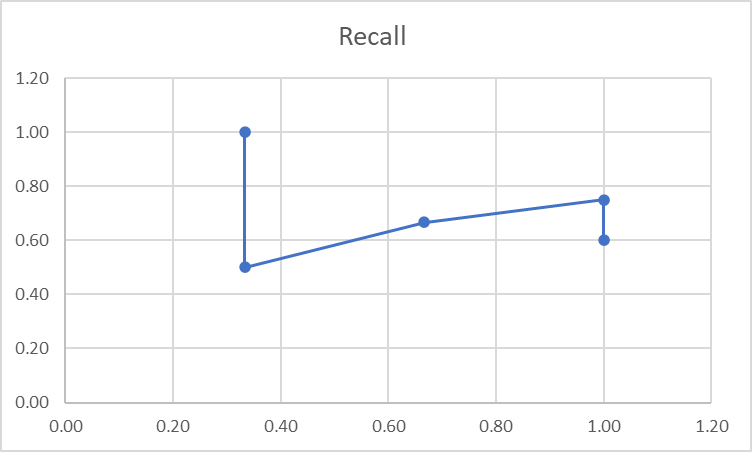
3.a.iv)

From the tutorial, we need to calcuate these values at different thresholds. Because these values have been returned in order of relevance, for the sake of argument, suppose the values are returned with similarity of: 90%, 80%, 70%, 60%, 50%.

,

At 90% confidence, only the first would be returned from the model. Because we’ve confirmed that it is +1, then it is a TP. All the other outputs are classified as not accurate enough to output. Therefore, -1 = TN as it appears below our confidence threshold and it was correctly classified as being ‘not good enough to output’ +1 = it should have been output with our 90% confidence, but we weren’t confident enough, so it’s a FN.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Threshold | TP | TN | FP | FN | Precision | Recall |
| 90% | 1 | 2 |  | 2 | 1 | 1/3 |
| 80% | 1 | 1 | 1 | 2 | 1/2 | 1/3 |
| 70% | 2 | 1 | 1 | 1 | 2/3 | 2/3 |
| 60% | 3 | 1 | 1 | 0 | 3/4 | 1 |
| 50% | 3 | 0 | 2 | 0 | 3/5 | 1 |



3.a.v

Apply preprocess to the input image, and use pretrained network (ResNet, VggNet, BeiT, DeiT)/algorithm (HoG) to get the image embeddings, then use Approximate Nearest Neighbor (e.g. KD-Tree) instead of KNN to do the search.

4.a.i)

This Siamese network performs object tracking by using a convolutional network (𝜑) to compare the exemplar image (𝑧) with the search image (𝑥) at an arbitrary time frame. The convolutional network is used to extract features from both images and then a cross-correlation operator (\*) is applied to find the similarity between the two images. If the similarity is above a certain threshold, then the object in the search image is determined to be the same as the object in the exemplar image, and thus the object has been tracked.

4.a.ii)

The convolutional network 𝜑 can be trained using labeled images, such as images of objects annotated with their locations, as well as video frames annotated with the same objects in different time frames. In addition, the network can be trained using synthetic data generated from a simulation environment.

4.a.iii)

N: width/height, p: padding, k: kernel size, s: stride, assume d: dilation = 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Layer | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Name | Input | conv1 | pool1 | conv2 | pool2 | conv3 | conv4 |
| Input | - | Input | conv1 | pool1 | conv2 | pool2 | conv3 |
| Filter Shape | - | 11x11x3 | 2x2 | 5x5x96 | 2x2 | 3x3x256 | 3x3x192 |
| Number of Filters | - | 96 | - | 256 | - | 192 | 128 |
| Stride | - | 2 | 2 | 1 | 2 | 1 | 1 |
| Padding | - | Same | - | Same | - | Same | Same |
| Feature Map Shape | 128x128x3 | 64x64x96 | 32x32x96 | 32x32x256 | 16x16x256 | 16x16x192 | 16x16x128 |
| Receptive Field | 1x1 | 11x11 | 13x13 | ~~21x21~~  29x29 | ~~21x21~~  33x33 | ~~25x25~~  49x49 | ~~27x27~~  65x65 |

4.a.iv)

We can do this by training the network 𝜑 with different scale of input. For example, the network can be trained using labeled images that have been resized to different scales or using video frames that have been resized to different scales. Additionally, the network can be trained using synthetic data generated from a simulation environment that includes objects of varying sizes.