Image Processing III
Mutual Information

Aadil Anil Kumar
Otmane Sabir

26/4/2020

Introduction

The third homework assignment required us to implement the similarity metric algorithm "mutual information" while following certain guidelines which could be summarized to the following list:

1. Implement the algorithm.
2. Choose 3 images from the internet and separate the green and red channels as two separate gray scale images each. Crop the green channel by cutting 20 pixels from the left and right sides, respectively - resulting in a cropped green channel image with 40 pixels smaller than the red channel image. Then virtually move the red channel images in x-direction over the corresponding green channel image in 41 steps from the left to right, and compute the mutual information of the overlapping regions for every step.
3. Plot the mutual information as a function of the x-position of the red channel image for all three chosen red/green image pairs.

Contents

1 Mutual Information

Mutual information is a quantity that measures a relationship between two random variables that are sampled simultaneously. In particular, it measures how much information is communicated, on average, in one random variable about another. For example, suppose X represents the roll of a fair 6-sided die, and Y represents whether the roll is even (0 if even, 1 if odd). Clearly, the value of Y tells us something about the value of X and vice versa. That is, these variables share mutual information. [1]

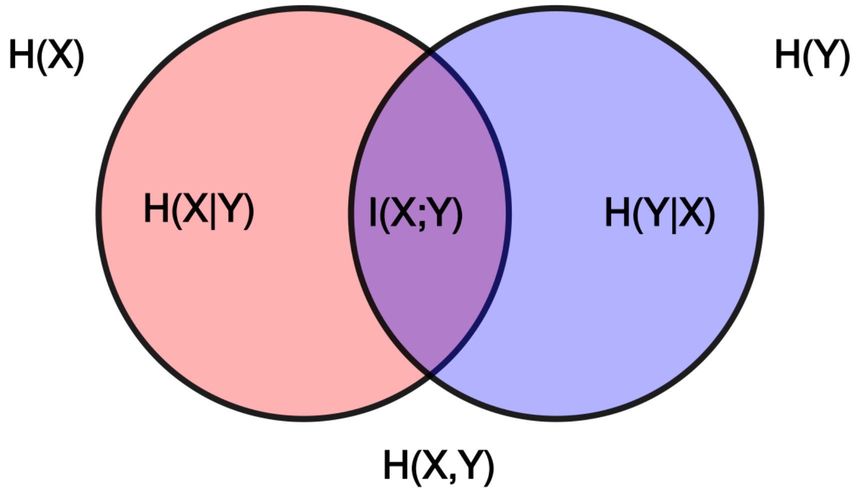

Figure 1: Mutual Information Visualization

1.1 Formal Definition

DEFINITION:[2][3]

The formal definition of the mutual information of two random variables X and Y, whose joint distribution is defined by $P(X, Y)$ is given by

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}$$

Such that, $P(X)$ and $P(Y)$ are the *marginal distributions* of X and Y obtained through the marginalization process which can be defined as :

$$p_X(x_i) = \sum_j p(x_i, y_i)$$

$$p_Y(y_j) = \sum_i p(x_i, y_i)$$

2 Shannon Entropy & Mutual Information

In our implementation, we rely on calculating the shannon entropy in order to estimate the mutual information. Intuitively, some may ask what is entropy?

The entropy is the expected value of the self-information, the self-information quantifies the level of information or surprise associated with one particular outcome or event of a random variable, whereas the entropy quantifies how "informative" or "surprising" the entire random variable is, averaged on all its possible outcomes.

2.1 Formal Definition : Shannon Entropy

DEFINITION:[4]

The formal definition of the entropy of a discrete random variable X with possible values $\{x_1, ..., x_n\}$ and probability mass function $P(X)$ is

$$H(X) = -\sum P(x_i) \log_b P(x_i)$$

where b is the base of the logarithm used. Common values of b are 2, Euler's number e, and 10. In the case of $P(x_i) = 0$ for some i, the value of the corresponding summand $0 \log_b(0)$ is taken to be 0.

2.2 Formal definition : Mutual Information

DEFINITION:[3]

If we consider pairs of discrete random variables (X,Y), then formally, the mutual information can be defined as :

$$I(X : Y) = H(X) + H(Y) - H(XY)$$

with $H(X)$, $H(Y)$ the Shannon entropy of X an Y, and $H(XY)$ the Shannon entropy of the pair (X, Y).

3 Implementation

We decided to implement our algorithm in python for ease of implementation and the support provided from preexisting libraries. We first started by implementing a simple entropy computation method, then by interpreting the image signals from the image to match our entropy function and then applying the formula from the previous definition.
Before we start implementing, this is a list of all libraries and styles used in order to calculate, convert and plot our results.

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import entropy as scipy_entropy
from PIL import Image
plt.style.use('seaborn')
```

3.1 Entropy function :

This is a simple implementation of the entropy function. Given the input parameter X which in our case is a random variable, we get the entropy of the random variable. In subsection **2.1** we introduced a formal definition of this calculation.

```python
def entropyCalc(X):
    uniq = set(X)
    P = [np.mean(X == x) for x in uniq]
    return sum(-p * np.log2(p) for p in P)
```

Listing 1: Entropy Calculator

3.2 Converting the image

In order for our entropy function to properly compute these values we need to convert the image from the matrices we receive to appropriate signals. We do this by using the numpy histogram functions in the case of the join histogram.

```python
def convert1DSignal(x, nBins):
    return np.histogram(np.asarray(x).flatten(), bins=nBins)[0]
```

Listing 2: 1D Signal

```python
def convert2DSignal(x, y, nBins):
    hist = np.histogram2d(np.asarray(x).flatten(), np.asarray(y).
    flatten(), bins=nBins)
    return np.asarray(Image.fromarray(hist[0], 'RGB')).flatten()
```

Listing 3: 2D Signal (Joint Histogram)

3.3 Mutual Information

We now have all the required tools to calculate our mutual information estimation. The mutual information function will be very simple and will follow our definition in the subsection **2.2**.

```python
def mutualInformation(x, y, nBins):
    HX = entropyCalc(convert1DSignal(x, nBins))
    HY = entropyCalc(convert1DSignal(y, nBins))
    HXY = entropyCalc(convert2DSignal(x, y, nBins))
    return HX + HY - HXY
```

Listing 4: Mutual Information

4 Experiments & Results

The experiments we were asked to conduct were an intuitive way of visualizing how mutual information exactly works. We were asked to choose 3 images from the internet and separate the green and red channels as two separate gray scale images each. Crop the green channel by cutting 20 pixels from the left and right sides, respectively - resulting in a cropped green channel image with 40 pixels smaller than the red channel image. Then virtually move the red channel images in x-direction over the corresponding green channel image in 41 steps from the left to right, and compute the mutual information of the overlapping regions for every step and then finally plot the mutual information as a function of the x-position of the red channel image for all three chosen red/green image pairs.

Before we proceed, we want to clarify the difference between bin size and number of bins. Given that x is the bin size we want, that y is the number of bins and that the image is a gray scale image - meaning the maximum number of bins is 256 - then $y = 256/x$ or $x = 256/y$ which will explain the nature of the varying number of bins plots (See figures 4, 7 & 10.)

4.1 Test function implementation

In order to achieve this we had to create a function which would trim the given image from the sides and create all other possible crops since our mutual information can't take signals of different sizes as it will be comparing a given value to nothing which would always return an error; therefore, our approach was to create a symmetrically cropped image from as the top image and create a list of all possible cropped images starting from 40 pixels to right to 40 to the left.

```python
def cropImage(img):
    r, g, b = splitImage(img)
    w, h = img.size
    top = r.crop((20, 0, w-20, h))
    return top, [g.crop((40-x, 0, w-x, h)) for x in range(40, -1,
    -1)]
```

Listing 5: Mutual Information

4.2 Puffin Experiment

For the first experiment, we decided to use a 800X800 puffin image (See Figure 2). We then ran our test functions with different bin sizes and with image translations.

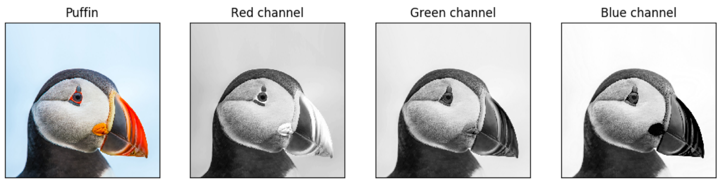

Figure 2: Original Puffin Image & Splits.

The results we got were as expected (see Figure 3). The mutual information value peaks when the the image translations is at 20 as it has reached the optimal combination of the red and green channels since both images are now symmetrically cropped (20 pixels from left and 20 from the right).

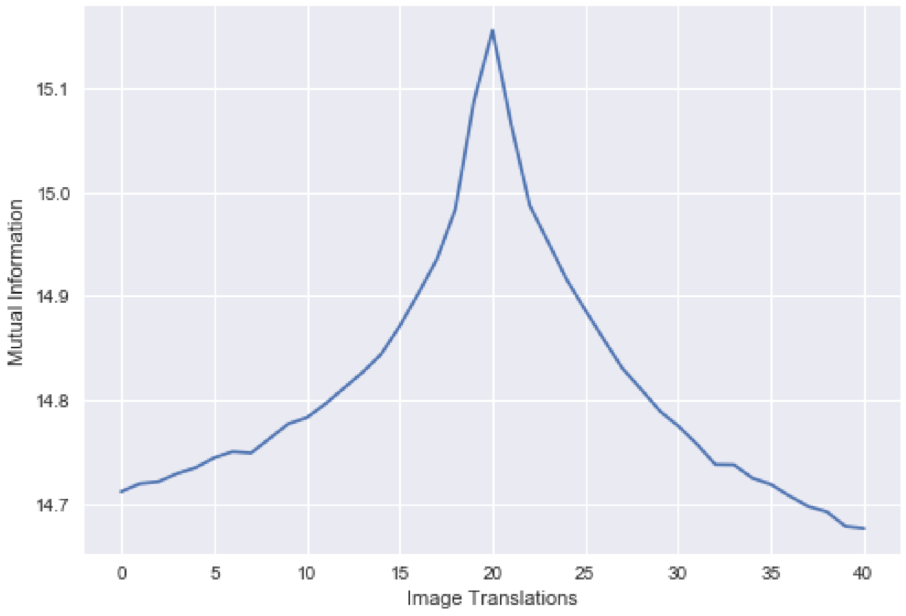

Figure 3: Translation Test with MI

The results we got were as expected (see Figure 4). The mutual information approximation drops as we increase the bin size as it becomes less and less precise the higher our bin size gets until it completely flattens out and finds no mutual information.

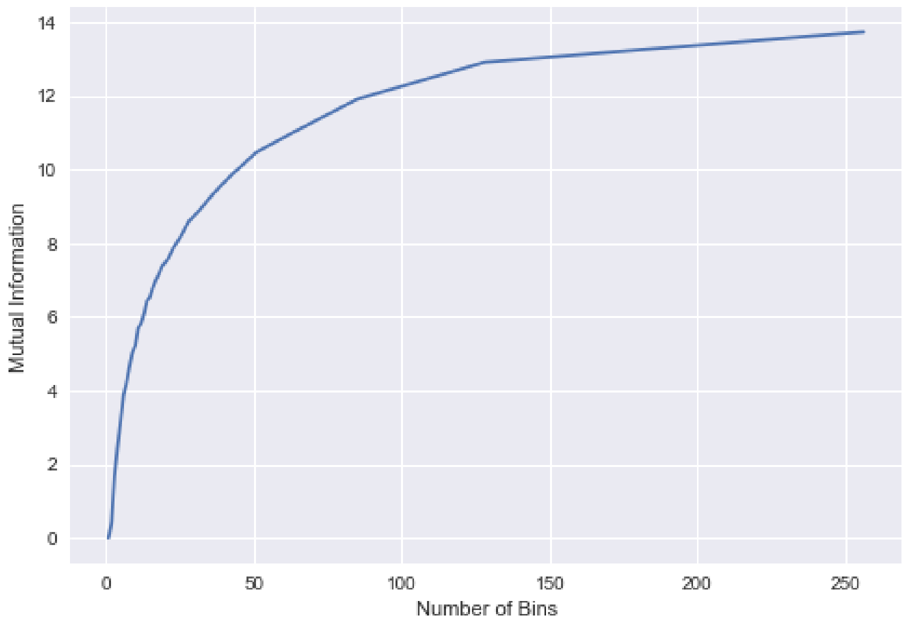

Figure 4: Bin Size Change - Puffin

4.3 Balloon Experiment

For the second experiment, we decided to use a 1024X694 balloon image (See Figure 5). We then ran our test functions with different bin sizes and with image translations.

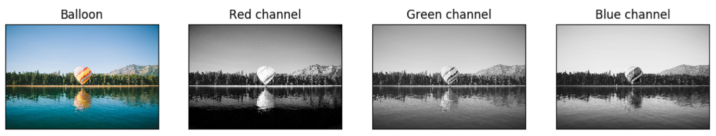

Figure 5: Original Balloon Image & Splits.

While the results were not as steady as the previous experiment but these also allow us to interpret the function from a different point a view and open a new level of understanding. As we can see in the original image splits (Figure 5). The resulting gray scale images are quite different from each other allowing more room for fluctuation and less mutual information which directly translates to the result we got in Figure 6 since we can't exactly predict how the red and green channels look like during the translation but it remains correct that the mutual information peaks when it reaches the symmetric crop.

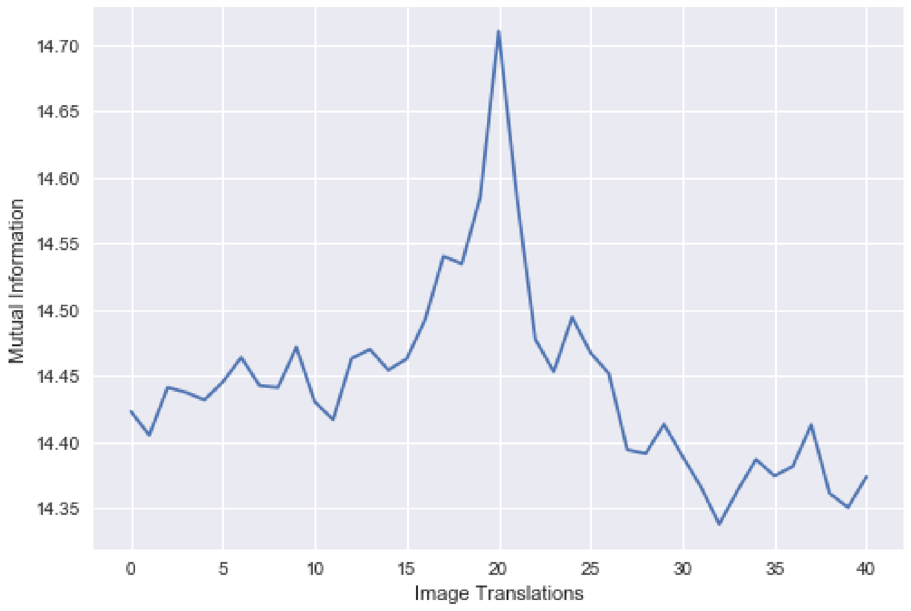

Figure 6: Translation Test with MI - Balloon

We can see that the results are similar to the previous one. As the number of bins increases, the mutual information increases since we, again, have more "information" than with previous images. It's also important to mention that the MI increases because the images are relatively similar and that only few gray scale values differ which explain why for this scenario the MI increase the higher the number of bins.

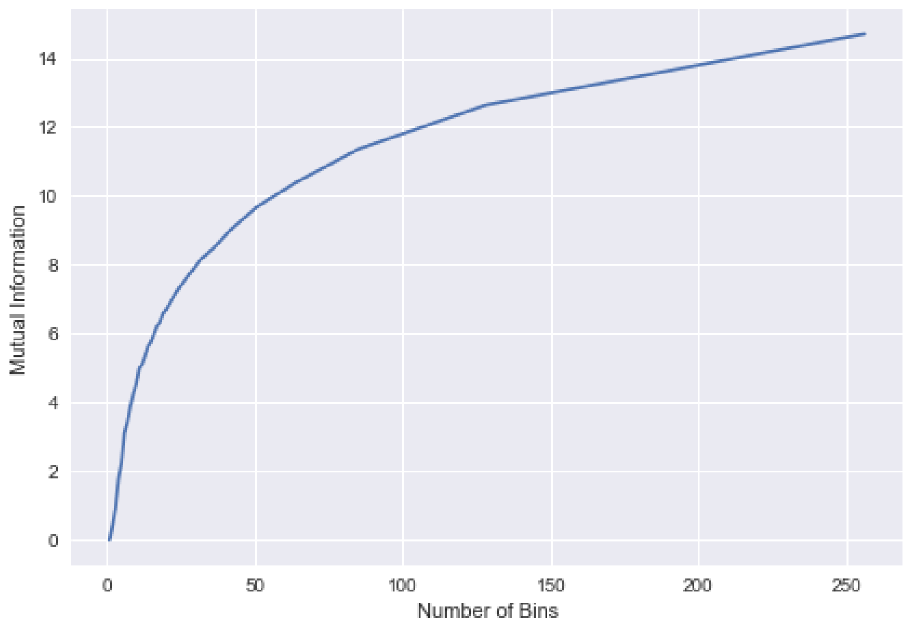

Figure 7: Bin Size Change - Balloon

4.4 Flower Experiment

For experiment 3 we used a 256x256 flower image (See Figure 2 and ran the same tests as in the two past experiments.

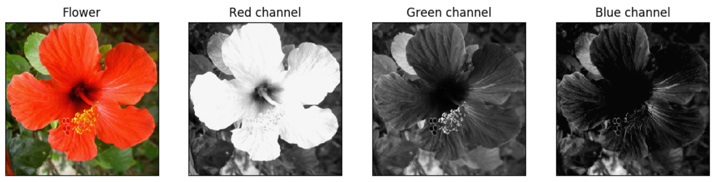

Figure 8: Original Flower Image & Splits.

The results showed a lot more fluctuation this time but kept the peak at the same place (Figure 9. We can first explain these by the big difference in the green and red gray scale images (See Figure 8. Another reason the fluctuations are higher could be the image resolutions since each pixel shift has a lot more ”weight”. For example, shifting by one column of pixels in a 800X800 image means we’re changing around 0.125% of the image while shifting by one column in a 256X256 image means we're changing 0.39% of the image which gives the shift more importance as the image is smaller.

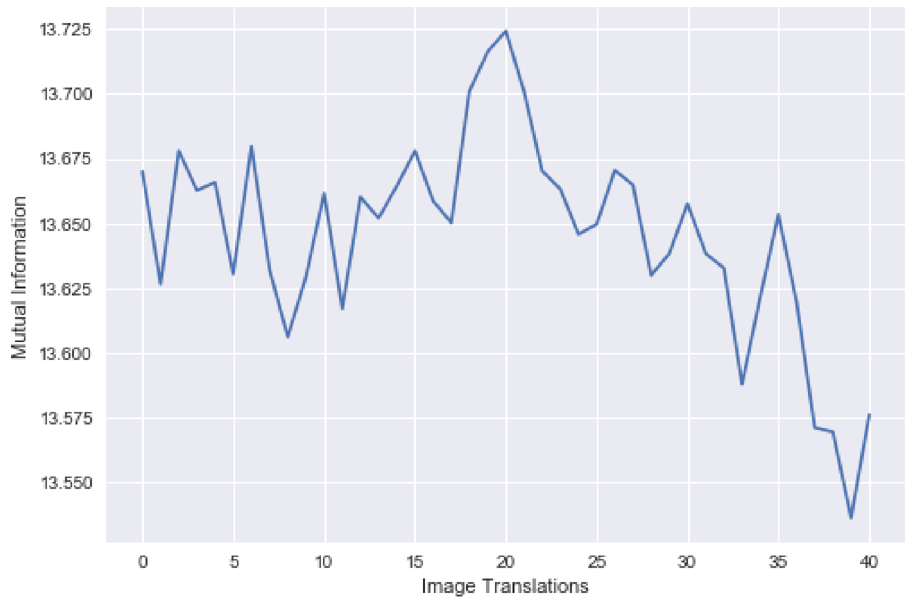

Figure 9: Translation Test with MI - Flower

The results were again similar to both previous experiments which confirm our explanation.

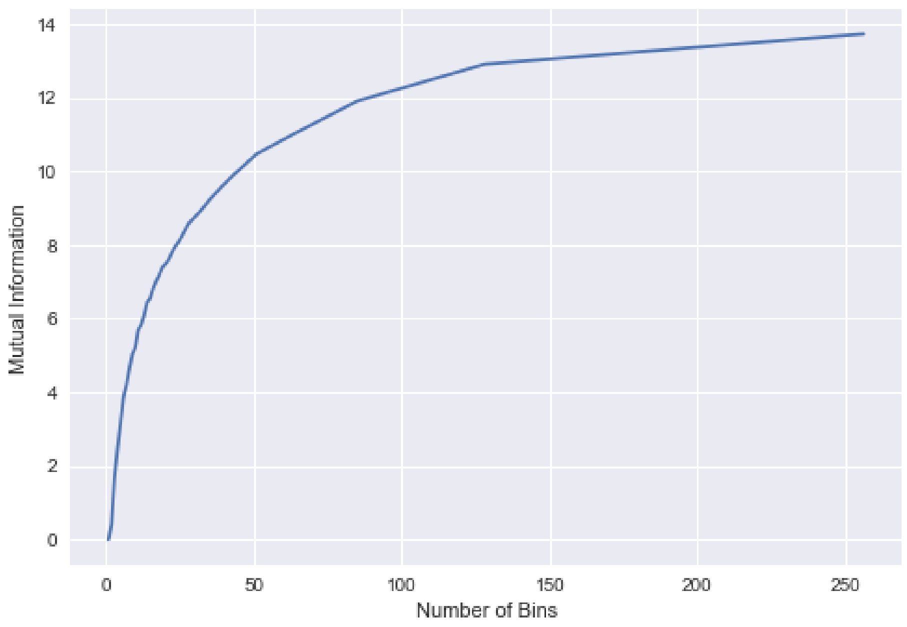

Figure 10: Bin Size Change - Flower

References

[1] Entropy and Mutual Information:
AS Kornillov
https://people.cs.umass.edu/ elm/Teaching/Docs/mutInf.pdf

[2] Mutual Information:
Wikipedia
https://en.wikipedia.org/wiki/Mutual$_{i}nformation$

[3] Mutual Information :
Quantiki :
https://www.quantiki.org/wiki/mutual-information

[4] Entropy (Information Theory) :
Wikipedia :
https://en.wikipedia.org/wiki/Entropy$_{(}information_{t}heory)$